MANUEL TECHNIQUE DES TO8, TO9 ET TO9+

Jean-Claude Mariaccia et Olivier Savin

cedic/nathan

Manuel technique des TO8, TO9, TO9+

Sommaire

Avant-propos 13

Première partie: Présentation des produits 15

Caractéristiques générales du TO9 17
Carte mémoire du TO9 19
Répertoire des différents registres 19
Caractéristiques générales du TO8 21
Carte mémoire du TO8 23
Caractéristiques générales du TO9+ 24
Carte mémoire du TO9+ 25
Répertoire des différents registres
concernant les unités centrales TO8, TO9+ 26

Deuxième partie: Analyse matérielle du TO9 29

1. Analyse générale 31
Conception générale 32

2. Le 6809 E dans le TO9 34
Principe fondamental 34
Interconnexion du 6809 E et de ses bus 35

3. Système de mémorisation 37
Les ROMS 37
- Commutation des slots 38
- Commutation des banques 38
- Fonctionnement d'ensemble 39
- Routines de commutation 39

Les RAMS 39
- Fonctionnement d'une 4416 40
- Principe fondamental 40
- Forme de l'adressage 42
- Organisation mémoire vive du TO9 42
- Sélections 42
- Lecture-écriture des RAMS 44
- Routine de commutation de banques 44

4. La gestion du système 46
Les décodages d'adresse 46
La génération des fonctions 47
Le gate array système 48
Description fonctionnelle 48

5. Le système de visualisation du TO9 52
Généralités 52
Construction globale de l'écran 52
Construction de la fenêtre active 53
Configuration de base 54
Gate array d'affichage et modes d'affichage 54
Rôle du circuit I-27 54
Description du circuit 55
Les différents modes d'affichage 57
Circuit de palette 67
Description fonctionnelle de l'IGV 68
Programmation de la palette 69
Exemple de routine simple 70
Circuit d'incrustation 71

6. Les interfaces parallèles 73
Utilisation du 6846 dans le TO9 73
Description fonctionnelle 73
Adresses des registres internes 74
Utilisation du 6821 dans le TO9 75
Description fonctionnelle 75
Adresses des registres internes 76

7. La gestion du clavier et des périphériques 77
Utilisation du 6850 dans le TO9 77
Description 77
Le clavier 79
Présentation 79
Signaux échangés avec l'unité centrale 80

8. Gestion du crayon optique 82
Fonctionnement du crayon optique 82
Fonctionnement de l'interrupteur 83
Fonctionnement du phototransistor 83

9. L'exploitation du lecteur-enregistreur de disquettes 85
Description 85
Fonctionnement général 87

Troisième partie: Analyse matérielle du TO8 89

1. Analyse générale 91
Conception générale - Description 92

2. Le 6809 E dans le TO8 95

3. Gestion de la mémoire morte 96
Description des logiciels 96
Commutation des logiciels 96
Sélection d'une page moniteur 97
Sélection entre logiciels résidents et cartouche 98
Sélection des quatre banques de logiciels internes 98
Synthèse de fonctionnement 99

4. Les mémoires vives 100
Fonctionnement d'une 41256 101
Organisation générale 101
Sélections et correspondances 102
Ecriture et lecture des RAMS 103

5. Le gate array mode page 105
Définition du mode page 105
Gestion de la mémoire vive 106
Structure du circuit 106
Traitement des signaux multiplexés 108
Les registres de traitement 110
Description et programmation des registres accessibles en écriture 110
Description des registres accesssibles en lecture pour D0 = 0 114
Description des registres accesssibles en lecture pour D0 = 1 115

6. Le gate array mode page dans le TO8 117
Organisation du registre de traitement "système 1" 117
Diagramme des signaux multiplexés 118
Association entre adressage physique et adressage logique 119
Espace "cartouche" 119
Espace "écran" 120
Espace "système" 121
Espace "données" 121
Gestion de l'affichage 124
Gestion des couleurs du cadre 125
Les décodages d'adresses 126
Sélection de l'espace moniteur 126
Sélection de l'espace cartouche 126

Sélection de la zone des périphériques externes 127
Sélection du contrôleur de drive 127
Tableau récapitulatif 128
Gestion du crayon optique 128

7. Chaîne de visualisation 130

8. Les interfaces 132
Le 6846 132
Partie ROM 132
Partie PIA 133
Partie TIMER 134
Adresses des registres internes 134
Le 6821 système 134
Le 6821 Musique et jeux 135
Description des broches 135
Adresses des registres internes 137
Le 6804 138
Interfaçage du clavier 139
Fonctionnement 139

9. Le contrôleur d'unités de disquettes 141
Branchements du THMFC1 141
Description et programmation des registres 143
Spécification d'un secteur 146

Quatrième partie: Analyse matérielle du TO9+ 147

1. Conception générale - Description 150
2. Extension intégrée 156

Cinquième partie: Le moniteur 157

1. Généralités 159

2. Gestion alphanumérique de l'écran 162
Générateurs de caractères alphanumériques 162
Alphabet standard G0 162
Alphabet G2 163
Caractères utilisateurs 163
Affichage des caractères alphanumériques 163
Positionnement des caractères 165
Programmation du curseur 165
Détermination de la fenêtre de travail 167
Retour du curseur coin gauche 169

Descente d'une ligne 169
Remontée d'une ligne 169
Retour au début de ligne 169
Effacements divers 170
Effacement de la fenêtre 170
Extinction et allumage du curseur 170
Effacement d'une ligne 170
Affichages particuliers 171
Caractères accentués, alphabet G2 171
Caractères Télétel 173
Séquences d'échappement 175
Programmation des couleurs 176
Programmation des modes d'affichage 178
Dimension des caractères 179
Traitements divers 180
Affichage alphanumérique par la routine PLOT 181

3. Gestion graphique de l'écran 183
Mémorisation en RAM forme et couleur 183
Commutation couleur 183
Commutation forme 183
Allumage ou extinction d'un point graphique 183
Tracé d'un segment de droite 186
Dessiner avec des caractères 188

4. Lecture de l'écran 190
Lecture d'un point graphique 190
Lecture d'un caractère 191
Caractère normal 191
Minuscule accentuée ou c cédille 191
Caractère de l'alphabet G2 191

5. Gestion du clavier 193
Lecture rapide du clavier 193
Décodage du clavier 194
Programmation du clavier 196
Périphériques du clavier 197
Test des boutons du périphérique 198
Lecture du périphérique 198

6. Gestion du light pen 200
Test du switch light pen 200
Lecture de la zone pointée 200

7. Gestion des manettes de jeux 202

8. Gestion de l'interface de communication 203

9. Gestion du lecteur-enregistreur de cassettes 206

10. Contrôleur de disquettes 207
Gestion physique 207
Format BASIC MICROSOFT 209
Table d'allocation des fichiers 210
Le catalogue 210

11. Programmation de la palette 212
Ecriture-lecture d'un registre de couleurs 213
Programmation complète de la palette 215
Correspondance mode d'affichage-registres de couleurs 216
Fichier PALETTE-CFG 218

12. Génération de sons 219
Création d'un bip 219
Création musicale 219

13. Commutation des mémoires ROM 222

14. Accès à l'extramoniteur 224

15. Gestion des interruptions 225

16. Initialisation 227

17. Informations complémentaires 228
Points d'entrée standard du moniteur 228
Registres du moniteur (page 0) 229

Sixième partie: L'extramoniteur 233

1. Généralités 235
Principe de base 235
Initialisation de l'extramon 236

2. Le graphique 238
Généralités 238
La fenêtre de travail 238
Choix du type de tracé 238
Tracé d'un point 239
Tracé de droites 241
Motif de remplissage 242
Tracé de rectangles 242
Tracé d'ellipses 244

Remplissage d'une zone 246
Le micro-interpréteur graphique MIG 247
Codage et décodage d'images 253

3. Les tortues 255
Généralités 255
Initialisation 255
La visibilité 255
Le déplacement 256
La direction 256
La rotation 257
La taille 257
La trace 258
La vitesse 258
Le positionnement 259
La compilation d'une forme 260
Exemple d'une tortue en mouvement 261

4. Les mathématiques 263
Généralités 263
Description des accumulateurs 263
Echanges mémoires et accumulateurs 265
Liste des fonctions mathématiques 265

5. Le DOS 269
Initialisation du DOS 269
Cache disque 270
Ouverture d'un fichier 271
Lecture d'un caractère 272
Ecriture d'un caractère 272
L'accès direct 272
Fermeture d'un fichier 273
Lecture du catalogue 273
Lecture du nom d'une disquette 274
Backup d'une disquette 274
Lecteur source différent du lecteur destination 274
Lecteur source égal lecteur destination 274
Fiche de routines 275
Copie d'un fichier 275
Destruction d'un fichier 276
Changement de nom d'un fichier 277
Initialisation d'une disquette 277
Place libre sur une disquette 277
Taille d'un fichier 278
Numéro d'enregistremant courant 278
Exemple d'utilisation 279

6. L'éditeur 280

7. L'interpréteur musical 282

8. Les messages d'erreurs en anglais 283

9. Le DOS ICONIQUE 285
Généralités 285
Sélection d'un fichier 285
Saisie d'un nom de fichier 286
Sélection du lecteur courant 287
Appel au DOS ICONIQUE 288

10. Informations complémentaires 289
Extramon sous BASIC 512 289
Les numéros de fonction ou routine d'extramon 289
Les equates d'extramon 291

Annexe : La connectique 295

Avant-propos

Cet ouvrage propose une étude matérielle et logicielle approfondie des trois unités centrales TO9, TO8, TO9+.

La première partie présente les caractéristiques des machines. Le lecteur trouvera plus particulièrement le détail des cartes mémoires et les adresses des registres programmables, indispensables à tous ceux qui veulent s'aventurer dans les différents recoins de leur micro-ordinateur.

Les seconde, troisième et quatrième parties traitent de l'étude matérielle de chaque produit. Le sujet est systématiquement abordé à partir d'un synoptique représentant les différents circuits. Chaque élément du synoptique est ensuite repris et traité en particulier. A l'aide de schémas à caractère didactique, toutes les fonctions "vitales" sont exposées dans un esprit d'analyse et de synthèse qui devrait permettre facilement de comprendre leur fonctionnement (c'est du moins le vœu des auteurs).

L'analyse logicielle est exposée dans les cinquième et sixième partie, abordant respectivement l'étude du moniteur et de l'extramoniteur. Plusieurs chapitres détaillent les fonctions spécifiques telles que la gestion alphanumérique ou graphique de l'écran, la scrutation du clavier, etc.

Les routines sont présentées chacune sous forme de fiche technique de programmation, indiquant son nom, le point d'entrée, les registres ou paramètres d'entrée et de sortie, ainsi que son rôle. Cette fiche comporte souvent un programme de démonstration ou un exemple d'utilisation.

Ainsi, nous espérons que les fanatiques de la programmation en assembleur, toujours à l'affut d'informations techniques "croustillantes", trouveront matière à réflexion. De même, les acharnés du fer à souder désirant construire des interfaces en tout genre pourront puiser dans ce livre des informations utiles à leur étude (une annexe traitant de la connectique leur est spécialement destinée).

Enfin, à tous les curieux de nature qui veulent, par principe, mieux connaître leur machine, nous souhaitons une bonne lecture.

Première partie

Présentation des produits

Les trois appareils THOMSON cités dans cet ouvrage, bien que dissemblables, restent très proches par certains aspects. Ce livre mène l'étude matérielle des trois produits selon un ordre chronologique correspondant à leur apparition. Le TO9, premier du nombre, sert logiquement de référence au TO8 ; le TO9+, dont la technologie découle du TO8, vient s'inscrire, quant à lui, en comparaison avec ses deux aînés. Ainsi, dans les troisième et quatrième parties concernant le TO8 et le TO9+, nous invitons constamment le lecteur à se reporter en "amont" aux parties traitant d'un fonctionnement identique.

Caractéristiques générales du TO9

Succédant au TO7/70, le TO9 est apparu en 1985 comme un ordinateur familial haut de gamme. Aux possibilités multiples, il s'est plus particulièrement affirmé au niveau du graphisme. Dans la taille des "huit bits", il figure comme un des meilleurs micro-ordinateurs de sa génération.

La description ci-après passe en revue la technologie et les différentes caractéristiques de l'unité centrale :

• Microprocesseur utilisé : 6809 E d'origine MOTOROLA EFCIS.

• Organisation mémoire morte : 136 Ko de ROM dont
– 8 Ko pour les moniteurs (système et lecteur de disquettes)
– 32 Ko pour le BASIC 128 et l'EXTRA-MONITEUR (slot 0 banques 0 et 1),
– 32 Ko pour le BASIC 1, la page d'en-tête, réglage de la palette et le DOS ICONIQUE (slot 0 banques 2 et 3),
– 2 × 32 Ko pour les progiciels PARAGRAPHE et FICHES ET DOSSIERS (slots 1 et 2),
– 16 ou 64 Ko sont réservés à une cartouche extérieure (slot 3).

• Organisation mémoire vive : 128 Ko de RAM dont:
– 16 Ko pour la partie écran et 16 Ko utilisateur non commutables,
– 96 Ko en 6 banques utilisateur de 16 Ko chacune et commutables (banques 0 à 5),
– 64 Ko peuvent être ajoutés comme extension (application en disque virtuel).

• Utilisation de trois "gate array" :

– un gate array pour les décodages d'adresses,
– un gate array principal pour la gestion générale du système, dont l'exploitation du crayon optique,

– un gate array pour les huit modes d'affichage :

mode TO7,
mode bit-map 4 couleurs,
mode 80 colonnes,
mode page 1,
mode page 2,
mode superposition 2 pages,
mode bit-map 16 couleurs,
mode superposition 4 plans.

Ces modes sont un compromis entre la définition de l'image et le nombre de couleurs, l'augmentation de l'un se faisant au détriment de l'autre.

• Utilisation d'un circuit de palette pour définir les 16 teintes exploitables parmi les 4 096 et d'un système d'interfaçage vidéo-son (prise SCART, incrustation, vidéo composite).

• Utilisation de trois circuits d'interface MOTOROLA EFCIS dont :
– un 6821,
– un 6846,
– un 6850,
permettant de gérer, plus particulièrement chacun, une imprimante en mode CENTRONICS, le LEP (magnétophone), le clavier. Dans le bloc clavier un monochip du type 6805 assure le transfert des informations séries apportées par les touches, par la souris ou par les paddles.

• Un circuit contrôleur de lecteur de disquettes assure la commande d'un lecteur-enregistreur de disquettes 3,5 pouces simple face, ainsi que d'une unité extérieure dont le standard peut être en 5,25 pouces et double face.

• Une série de trois connecteurs arrière permettent le branchement de trois extensions différentes (contrôleurs et interfaces).

• Un quatrième connecteur arrière est destiné à recevoir l'extension disque virtuel.

Carte mémoire du TO9

Adresses	Désignation	Commentaires
$0000-$3FFF	ROM(4 SLOTS)	Logiciels résidents et cartouche
$4000-$5FFF	RAM écran A	8 Ko partie A (forme)
	RAM écran B	8 Ko partie B (couleurs)
$6000-$60FF	RAM	Page 0 du moniteur
$6100-$9FFF	RAM	Non commutable
$A000-$DFFF	RAM données	Banques commutables de 6 à 10 (extension)
$E000-$E7AF	ROM moniteur	Partie lecteur de disquettes
$E7B0-$E7BF	Zone libre	Réalisations particulières
$E7C0-$E7C7	6846 système	PIA TIMER
$E7C8-$E7CB	6821 système	PIA interne
$E7CC-$E7CF	6821 jeux & musique	PIA externe
$E7D0-$E7D9	Contrôleur	Pour le lecteur de disquettes
$E7DA-$E7DD	Registres d'écran	Palette et mode d'affichage
$E7DE-$E7DF	6850 système	ACIA liaison clavier
$E7E0-$E7E3	Zone libre	Ancien contrôleur de communication non utilisable
$E7E4-$E7E7	Latches gate array	Gestion du crayon optique
$E7E8-$E7EB	Extension contrôleur de communicat.	ACIA pour standard RS232
$E7EC-$E7EF	Libres	
$E7F0-$E7F7	Extension IEEE	
$E7F8-$E7FF	Extension MODEM	
$E800-$FFFF	ROM moniteur	Partie unité centrale.

Répertoire des différents registres

Adresses	Registres	Equate
6846 système		
$E7C0	Etat composite	CSR
$E7C1	Contrôle port C	CRC
$E7C2	Direction port C	DDRC
$E7C3	Données port C	PRC
$E7C4	Etat composite	
$E7C5	Contrôle du TIMER	TCR
$E7C6	TIMER poids fort	TMSB
$E7C7	TIMER poids faible	TLSB

Adresses	Registres	Equate
6821 système		
$E7C8	Direction/données port A	PRA
$E7C9	Direction/données port B	PRB
$E7CA	Contrôle port A	CRA
$E7CB	Contrôle port B	CRB
6821 Extension jeux et musique		
$E7CC	Direction/données port A	PRA1
$E7CD	Direction/données port B	PRA2
$E7CE	Contrôle port A	CRA1
$E7CF	Contrôle port B	CRA2
Contrôleur de lecteur de disquettes		
$E7D0	Etat (lecture seule)	STR
	Commande (écriture seule)	CMDR
$E7D1	Pistes	TKR
$E7D2	Secteurs	SECR
$E7D3	Données	DR
Palette		
$E7DA	Données	PALETTE
$E7DB	Adresses	PALETTE + 1
Gate array d'affichage		
$E7DC	Modes d'affichage (écrit. seule)	LGAMOD
$E7DD	Couleurs du tour (écrit. seule)	LGTOU
6850 Système		
$E7DE	Contrôle (écriture seule)	SCR
	Etats (lecture seule)	SSDR
$E7DF	Transmission de données (écriture seule)	STDR
	Réception de données (lecture seule)	SRDR

Adresses	Registres	Equate
Gate array système (gestion crayon optique)		
$E7E4	Adresses A15-A8 (lect. seule) et contrôle d'interruption (écriture seule)	LGA4
$E7E5	Adresses A7-A0 (lect. seule)	LGA5
$E7E6	Informations complémentaires (lecture seule)	LGA6
$E7E7	Informations complémentaires (lecture seule)	LGA7
SY6551 RS232 Contrôleur de communications		
$E7E8	Transmission (écriture seule)	SIOTRANSM
	Réception (lecture seule)	SIORECEPT
$E7E9	Etats (lecture seule)	SIOSTATUS
	Progr. reset (écriture seule)	SIORESET
$E7EA	Commande (écriture seule)	SIOCMDE
$E7EB	Contrôle (écriture seule)	SIOCNTRL
MODEM extension		
$E7F8	Direction/données port A	PORTA
$E7F9	Contrôle port A	COMMA
$E7FA	Direction/données port B	PORTB
$E7FB	Contrôle port B	COMMB
$E7FC	Inutilisé	
$E7FD	Inutilisé	
$E7FE	Contrôle (écriture seule)	ACIAS
	Etats (lecture seule)	ACIAS
$E7FF	Transmission (écriture seule)	ACIAD
	Réception (lecture seule)	ACIAD

Caractéristiques générales du TO8

Destiné à remplacer son aîné, le TO7/70, ce nouveau micro-ordinateur familial milieu de gamme THOMSON, améliore ses performances grâce à une technologie avancée. Bien que de conception différente, il reprend en grande partie les caractéristiques du TO9.

Description et possibilités :

• Microprocesseur utilisé : 6809 E d'origine MOTOROLA EFCIS.

• Organisation mémoire morte : 80 Ko de ROM dont
– 16 Ko pour les moniteurs (système et lecteur de disquettes),
– 32 Ko pour le BASIC 512 et l'EXTRA-MONITEUR (banques 0 et 1),
– 32 Ko pour le BASIC 1, la page d'en-tête et le réglage de la palette (banques 2 et 3).
– 16 à 64 Ko sont réservés à une cartouche extérieure, par commutation de banques de 16 Ko internes à la cartouche.

• Organisation mémoire vive : 256 Ko de RAM interne dont
– 16 Ko pour la partie écran et 16 Ko utilisateur non commutables (page 0, page 1),
– 224 Ko en 14 banques utilisateur de 16 Ko chacune et commutables (pages 2 à 15),
– 256 Ko de RAM externe ou extension (pages 16 à 31).

• Utilisation d'un gate array "mode page" regroupant :
– les décodages d'adresses,
– la gestion générale du système et l'exploitation du crayon optique,
– les huit modes d'affichage tels que :

mode TO7,
mode bit-map 4 couleurs,
mode 80 colonnes,
mode page 1,
mode page 2,
mode superposition 2 pages,
mode bit-map 16 couleurs,
mode superposition 4 plans.

Ce nouveau circuit offre la possibilité d'affecter les pages de RAM 1, 2, 3 en affichage vidéo (remplacement de la page 0). De même, il permet un recouvrement de l'espace ROM par les 32 pages de mémoire vive.

• Comme sur le TO9, utilisation d'un circuit de palette pour définir les 16 couleurs exploitables parmi les 4 096 teintes disponibles et d'un système d'interfaçage vidéo-son (prise SCART, incrustation, vidéo composite).

• Utilisation de trois circuits d'interface MOTOROLA EFCIS dont :
– deux 6821,
– un 6846,
permettant de gérer, plus particulièrement chacun, les manettes de jeux, la souris ou la synthèse sonore, une imprimante en mode CENTRONICS, le LEP (magnétophone), la liaison clavier.

• Utilisation d'un deuxième microprocesseur type 6804 afin d'interfacer le clavier avec le 6846 en mode de transmission série.

• Un deuxième gate array, spécialisé en circuit contrôleur de lecteur de disquettes, assure la commande d'un éventuel lecteur-enregistreur de disquettes externe, selon les trois standards :
5,25 pouces
3,5 pouces
QDD

• Une série de deux connecteurs arrière permet le branchement d'une extension telle que modem ou RS232 (connecteur extension polyvalent) ou bien l'extension mémoire vive.

Carte mémoire du TO8

Adresses	Désignation	Commentaires
$0000-$3FFF	ROM	Logiciels résidents et cartouche
$4000-$5F3F	RAM écran A	8 Ko partie A (forme)
	RAM écran B	8 Ko partie B (couleurs)
$5F40-$5FFF	RAM	Reste mémoire écran réservé au système
$6000-$60FF	RAM moniteur	Page 0 du moniteur
$6100-$9FFF	RAM	Non commutable
$A000-$DFFF	RAM données	14 banques ou pages commutables de 16 Ko ; 16 pages supplémentaires avec extension
$E000-$E7BF	ROM moniteur	2 pages de 1,9 Ko pour le lecteur de disquettes
$E7C0-$E7C7	6846 système	PIA TIMER
$E7C8-$E7CB	6821 système	PIA interne
$E7CC-$E7CF	6821 musique & jeux	PIA interne
$E7D0-$E7D7	Contrôleur	Pour le lecteur de disquettes
$E7D8-$E7D9	Sélection allouée au lecteur de disquettes	
$E7DA-$E7DD	Registres gate	Palette & mode d'affichage
$E7DE-$E7DF	Zone libre	Zone ACIA du TO9
$E7E0-$E7E3	Zone libre	Ancien contrôleur de communication non utilisé
$E7E4-$E7E7	Registres gate	Gestion mode page et crayon optique
$E7E8-$E7EB	Extension contrôleur de communicat.	ACIA pour standard RS232

Adresses	Désignation	Commentaires
$E7EC-$E7EF	Libre	
$E7F0-$E7F7	Extension IEEE	
$E7F8-$E7FF	Extension MODEM	
$E800-$FFFF	ROM moniteur	Partie unité centrale, en 2 pages de 6 Ko.

Un espace de FFD0 à FFEF est disponible pour des évolutions futures.

Caractéristiques générales du TO9+

Succédant au TO9, le TO9+ est une machine dont les caractéristiques bien qu'améliorées restent proches de son prédécesseur; mais une nouvelle technologie, dérivée du TO8, lui confère une souplesse et une fiabilité accrues.

Comme pour les systèmes précédents, la description suivante définit les différentes caractéristiques et l'architecture générale de l'unité centrale :

• Microprocesseur utilisé : 6809 E d'origine MOTOROLA EFCIS.

• Organisation mémoire morte : 80 Ko de ROM dont
– 16 Ko pour les moniteurs (système et lecteur de disquettes),
– 32 Ko pour le BASIC 512 et l'EXTRA-MONITEUR (banques 0 et 1),
– 32 Ko pour le BASIC 1, réglage de la palette, la page d'en-tête et le DOS ICONIQUE (banques 2 et 3),
– 16 à 64 Ko sont réservés à une cartouche extérieure.

• Organisation mémoire vive : 512 Ko de RAM dont
– 16 Ko pour la partie écran (page 0),
– 16 Ko système - utilisateur (page 1 fixe),
– 30 pages (2 à 32) de 16 Ko chacune et commutables.

• Utilisation du gate array "mode page" identique à celui du TO8 pour :
– les décodages d'adresses,
– la gestion générale de l'unité et l'exploitation du crayon optique,
– les modes d'affichage,
– les recouvrements.

• Utilisation d'un circuit de palette pour définir les 16 teintes parmi les 4 096 exploitables, et d'un système d'interfaçage vidéo-son (prise SCART, incrustation, vidéo composite).

• Utilisation de circuits d'interface MOTOROLA EFCIS (6821, 6846, 6850) pour :
– une imprimante du type CENTRONICS,
– des manettes de jeux ou la souris,

– la sortie son synthétisé, en association avec un convertisseur numérique-analogique,
– le LEP et le crayon optique,
– le transfert des informations entre le bloc clavier et la machine.

Dans le bloc clavier un monochip du type 6805 P2 assure le dialogue avec l'unité centrale.

• Utilisation d'un gate array contrôleur de lecteur de disquettes, pour gérer le lecteur-enregistreur de disquettes (FLOPPY) 3,5 pouces double face, double densité.

• Une extension télématique est incluse en version "France". Une extension RS232 est incluse en version "Export".

• Deux connecteurs arrière polyvalents permettent le branchement d'extensions différentes (contrôleurs et interfaces).

Carte mémoire du TO9+

Adresses	Désignation	Commentaires
$0000-$3FFF	ROM	Logiciels résidents et cartouche
$4000-$5F3F	RAM écran A	8 Ko partie A (forme)
	RAM écran B	8 Ko partie B (couleurs)
$6000-$60FF	RAM	Du moniteur
$6300-$9FFF	RAM	Fixe
$A000-$DFFF	RAM données	Banques ou pages commut.
$E000-$E7BF	ROM moniteur	Partie lecteur disquettes 2 × 1,9 Ko
$E7C0-$E7C7	6846 système	PIA TIMER
$E7C8-$E7CB	6821 système	PIA interne
$E7CC-$E7CF	6821 musique & jeux	PIA interne
$E7D0-$E7D9	Contrôleur	Pour le lecteur de disquettes
$E7DA-$E7DD	Registres d'écran	Palette et mode d'affichage
$E7DE-$E7DF	6850 système	ACIA gestion clavier
$E7E0-$E7E3	Zone libre	Ancien contrôleur de communication non utilisé
$E7E4-$E7E7	Gate array mode page	Gestion système et crayon optique
$E7E8-$E7EB	Extension contrôleur de communicat.	ACIA pour standard RS232 RS52932
$E7EC-$E7EF	Extension digitalisation	
$E7F0-$E7F7	Extension IEEE	
$E7F8-$E7FF	Extension MODEM	
$E800-$FFFF	ROM moniteur	Partie unité centrale. 2 × 6 Ko
$FFD0-FFEF	Espace disponible pour des évolutions futures	

Répertoire des différents registres concernant les unités centrales TO8, TO9+

Adresses	Registres	Equate
6846 système - cf. TO9		
6821 système - cf. TO9		
6821 Jeux et musique - cf. TO9		
Contrôleur de lecteur de disquettes		
$E7D0	Etat (lecture seule)	STATO
	Commande (écriture seule)	CMDO
$E7D1	Etat (lecture seule)	STAT1
	Commande (écriture seule)	CMD1
$E7D2	Commande (écriture seule)	CMD2
$E7D3	Données en écriture	WDATA
	Données en lecture	RDATA
$E7D4	Horloge en écriture	WCLK
$E7D5	Secteurs en écriture	WSECT
$E7D6	Pistes en écriture	WTRCK
$E7D7	Taille de la cellule "bit" en écriture	WCELL
Palette - cf. TO9		
Gate array mode page		
$E7DC	Modes d'affichage (écrit. seule)	LGAMOD
$E7DD	Système 2 (écrit. seule)	
6850 (TO9+) - cf. TO9		

Adresses	Registres	Equate
Gate array mode page		
$E7E4	Commutation (écrit. seule)	
	Crayon optique 1 (lect. seule)	
$E7E5	RAM données (écr. seule)	
	Crayon optique 2 (lect. seule)	
$E7E6	Espace cartouche (écrit. seule)	
	Crayon optique 3 (lect. seule)	
$E7E7	Système 1 (écrit. seule)	
	Crayon optique 4 (lect. seule)	
RS52932 (Contrôleur de communication) - cf. TO9		
MODEM - cf. TO9		

Deuxième partie

Analyse matérielle du TO9

1. Analyse générale

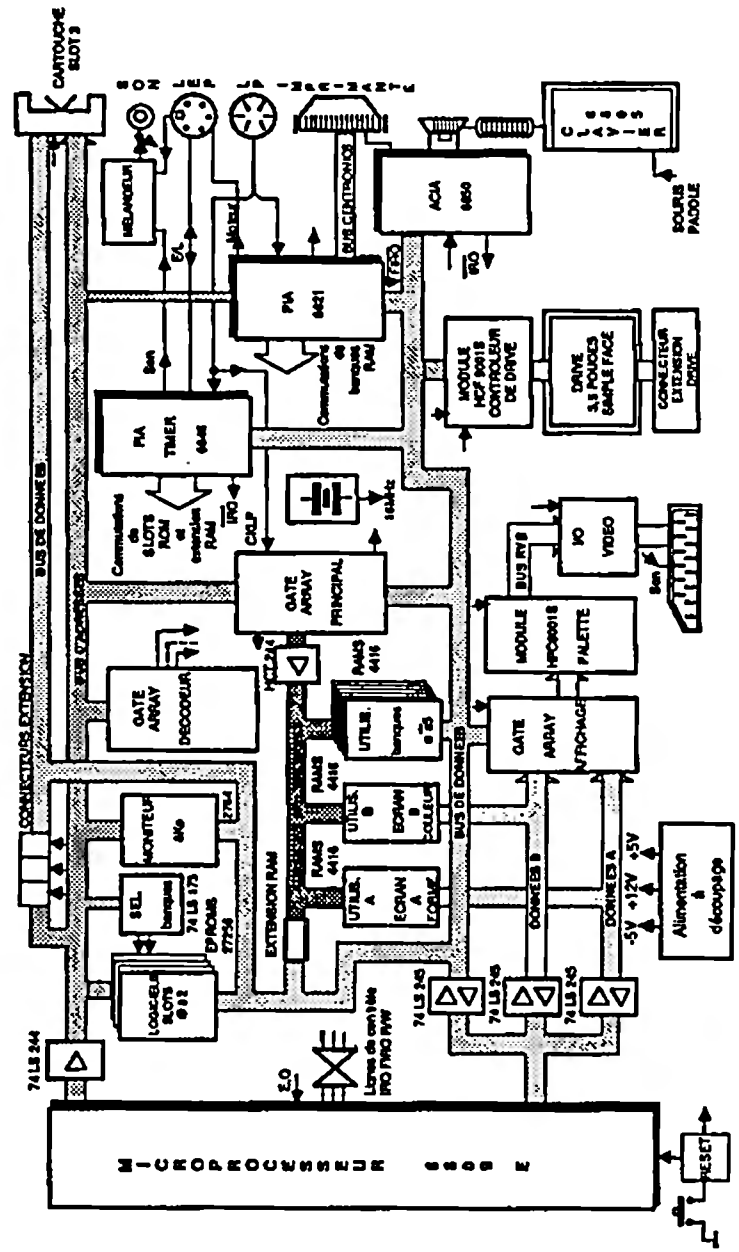


Figure 1. Synoptique de l'unité centrale TO9

Conception générale

Le micro-ordinateur TO9 est conçu autour d'un microprocesseur 6809 E, 8 bits recevant deux signaux d'horloge en quadrature (E et Q) à 1 MHz. Le bus d'adresses 16 bits bufferisé par des 74 LS 244 permet l'accès aux différentes mémoires et registres. Le bus de données 8 bits se répartit en trois groupes :

– Un bus de données principal 8 bits bufferisé et sélecté par un 74 LS 245, permettant les principaux échanges.

– Deux bus de données A et B de 8 bits, bufferisés et sélectés chacun par deux autres 74 LS 245 et permettant les transferts d'informations avec les deux groupes de RAMS, partie A, partie B.

Les lignes de contrôle sont constituées par les commandes de lecture-écriture (R/WN) des mémoires vives et de divers registres, ainsi que par les demandes d'interruption concernant le clignotement du curseur, le fonctionnement du clavier et du crayon optique (IRQN, FIRQN).

Un circuit de RESET en relation avec le 6809 E et différents registres permet la réinitialisation de la machine.

Les 6 Ko de logiciel de base, ou moniteur système, sont logés dans une EPROM 8 Ko 2764. Une partie des 2 Ko restant sert de moniteur au lecteur de disquettes.

Les 128 Ko de logiciels d'application résidents sont logés dans quatre EPROMS 32 Ko 27256, répartis en trois slots (0 à 2) de deux ou quatre banques.

La commutation des banques s'effectue en **programmation** par écriture d'adresses dans un LATCH 74 LS 173. Les slots sont eux-mêmes sélectionnés par deux bits de PIA en provenance du 6846.

La cartouche de logiciel d'application externe représente, en relation avec les bus de données, d'adresses et les lignes de contrôle, le slot numéro 3.

Les 128 Ko de mémoire vive répartis en huit groupes de deux 4416 (reconstitution des datas 8 bits) définissent la mémoire écran et utilisateur partie A, la mémoire écran et utilisateur partie B, ainsi que les six banques de RAM utilisateur. Chaque banque est commutée d'une façon logicielle, à partir de trois bits de PIA en provenance du 6821. L'extension mémoire est composée de quatre autres banques commutées par deux bits du PIA 6846 en association avec le 6821. Les 4416 étant des mémoires dynamiques, leur adressage ne provient pas en direct du 6809 E mais passe par un bus procurant un adressage multiplexé. Ce bus est bufferisé par un HCT244.

En dehors du microprocesseur, l'unité centrale est commandée par un gate array principal permettant, via le bus d'adresses multiplexées, le rafraîchissement des mémoires et de l'écran pendant la phase non active du 6809 E (technique de DMA). Attaqué par une horloge mère de 16 MHz, le circuit délivre les différents signaux d'horloge et de commande du téléviseur. Il assure aussi la gestion du crayon optique (LP).

Un deuxième gate array conçoit les différents décodages d'adresses inhérents à la machine. Le troisième gate array détermine, en relation avec le codage logiciel des RAMS écran partie A et partie B, les huit modes d'affichage. Ce circuit est programmable par le bus de données principal. Il reçoit les 16 bits d'informations des mémoires vives écran par les bus de données A et B.

Le module palette, programmable par le 6809 E en relation avec le bus de données 6809, permet, sous la dépendance du mode d'affichage considéré, un choix maximum de seize teintes parmi 4 096, pour la fenêtre écran et pour le tour ou cadre. Il délivre trois informations B, V, R reprises et adaptées par les circuits d'interfaçage vidéo comprenant, entre autres, le dispositif d'incrustation. Ces circuits, recevant des signaux de synchronisation et de blanking en provenance du gate array système, fournissent les tensions nécessaires, avec des impédances respectant les normes SCART, pour la prise Péritel. Une vidéo composite est reconstituée en sortie.

Le 6846 assure par son timer le clignotement du curseur (demandes IRQN) ou l'envoi codé des informations numériques à enregistrer sur le LEP (magnétophone). Inversement, son PIA récupère les informations de lecture décodées à charger. Une ligne du PIA génère le son qui, via un mélangeur recevant le son du LEP, attaque la prise Péritel et une fiche CINCH. Une autre ligne du PIA prend en compte l'information "tactile" du crayon optique LP (switch). Le restant du PIA contrôle certaines commutations de banques RAM, ainsi que des commutations de slots.

Le 6821, en dehors des commutations de banques RAM, a pour mission de prendre en compte les interruptions FIRQN sollicitées par le phototransistor du crayon optique et de commander, via un connecteur spécialisé, une imprimante en mode parallèle CENTRONICS. De même, il assure les demandes d'incrustation et la télécommande du moteur LEP.

A part le traitement du signal "BUSY" imprimante, le 6850 est entièrement consacré à la gestion du clavier. Travaillant en mode série asynchrone avec le monochip (type 6805) de ce dernier, il effectue des liaisons au rythme de 9 600 bauds.

Le lecteur de disquettes (drive) au standard 3,5 pouces simple face, simple ou double densité, est commandé par un module de contrôle (type WD 2793 ou WD 1770) programmable par le 6809. Le choix de la densité se fait par une opération de "latch" en écriture dans un circuit annexe 74 LS 173.

Une alimentation à découpage du type "fly back" à régulation opto-électronique fabrique les trois tensions de + 5 V, + 12 V, - 5 V, nécessaires à la configuration du micro ordinateur.

2. Le 6809 E dans le TO9

Principe fondamental

Depuis l'apparition du tout premier TO7, les unités centrales des micro-ordinateurs THOMSON ont un fonctionnement typique basé sur le même principe fondamental. Ce principe consiste à prendre en compte cycliquement la phase active et non active du 6809.

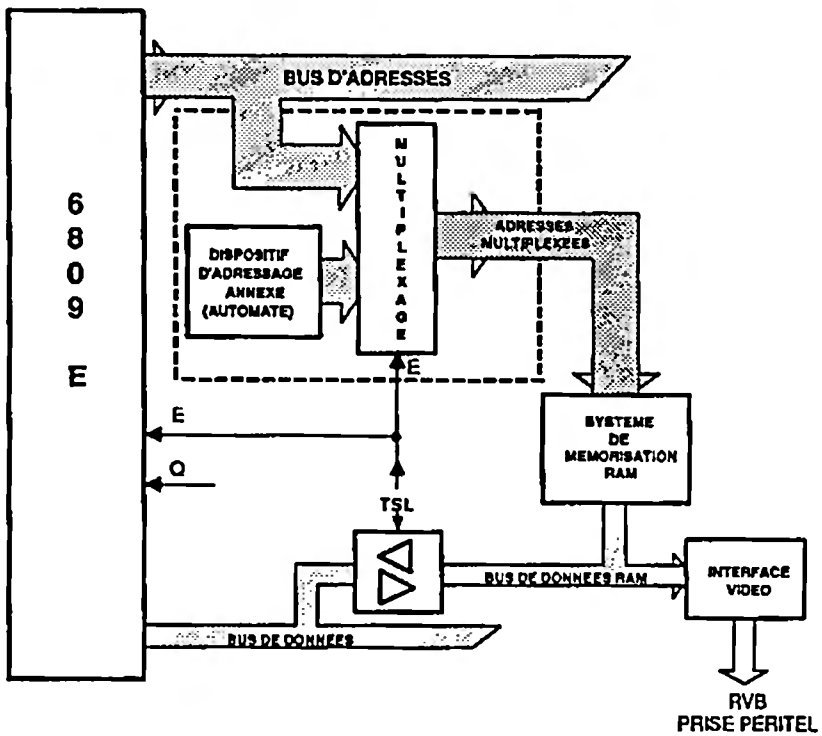


Figure 2. Dispositif fondamental dans les unités centrales Thomson 8 bits

Ainsi lorsque le signal d'horloge machine E = 1, le microprocesseur est dans sa phase active de fonctionnement; ce dernier adresse normalement la carte mémoire du système et réalise des échanges sur les bus de données en relation avec l'exécution de l'instruction en cours.

Lorsque E = 0, le microprocesseur est dans sa phase inactive de fonctionnement; un dispositif d'adressage annexe (automate) prend alors le relai afin d'assurer, via l'interface vidéo, le rafraîchissement de la mémoire écran et, simultanément par la même occasion, le rafraîchissement nécessaire à la technologie des mémoires vives utilisées (RAMS dynamiques). Le bus de données est déconnecté du bus de données RAM par l'intermédiaire d'un dispositif "3 états".

La commutation des adresses 6809 et des adresses du dispositif annexe s'effectue à partir d'un multiplexeur. Le signal E, alternativement à 1 et à 0 toutes les microsecondes, commande un adressage CPU et un rafraîchissement d'une durée conjointe de 500 nanosecondes (cf. figure 2).

Interconnexion du 6809 E et de ses bus

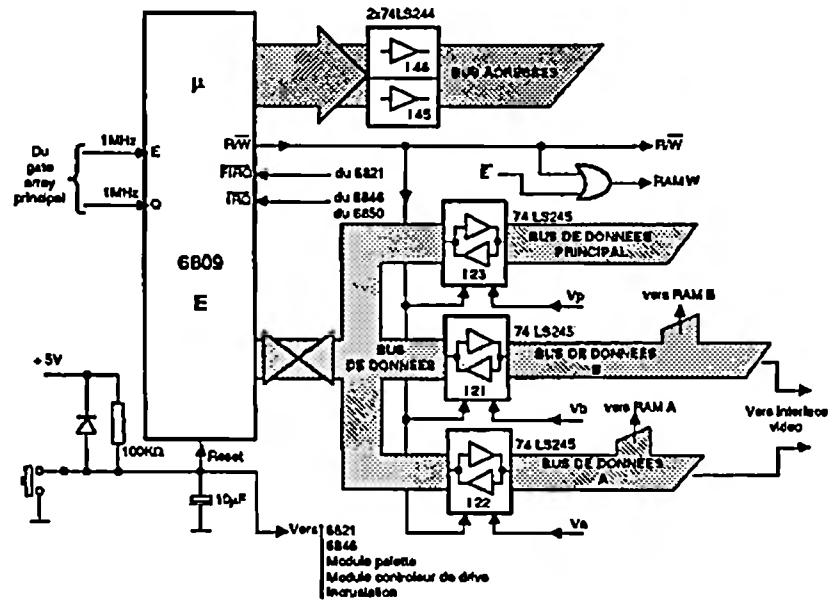


Figure 3. Le CPU et ses bus dans le TO9

La figure 3 montre quelles sont les entrées-sorties utilisées sur le microprocesseur.

La fréquence d'horloge E et Q en provenance du gate array principal est de 1MHz. Le circuit de RESET utilise une constante de temps de 0,00001 × 100000 = 1 seconde. Afin d'adapter la sortance, le bus d'adresses est bufferisé par deux circuits 74 LS 244.

Le bus de données se répartit en trois bus dont chacun est bufferisé par un 74 LS 245 commandé en bidirectionnel par le signal de lecture écriture R/WN.

Afin d'éviter tout conflit lorsque le 6809 est en lecture, un seul des 74 LS 245 est actif à la fois, les deux autres étant en état "haute impédance". Cette action est dévolue aux commandes de validation Vp, Vb, Va. Exception faite pour Vp, elles génèrent des signaux issus de circuits combinant les décodages d'adresses correspondants avec E et R/WN. Il en résulte pour Vb et Va des actions de validation particulières, pendant la phase non active du 6809, ou lorsque le microprocesseur est en écriture.

Ainsi lorsque E = 0, le bus de données A et le bus de données B ne sont plus en relation avec le bus de données du microprocesseur, les informations en mémoire vive étant transférées automatiquement vers le système d'interfaçage vidéo.

De même, pour E = 1 lorsque le 6809 E est en écriture, les bus A et B sont connectés au bus du microprocesseur, ce qui n'apporte pas de conflit puisque les buffers sont en entrée mais permet aux données d'être présentes en entrée des RAMS 4416 bien avant que celles-ci soient elles-mêmes validées (considérations de timing).

La commande R/WN du 6809 E est envoyée à tous les registres succeptibles de travailler en lecture-écriture dans l'unité centrale. Les circuits de RAM, quant à eux, sont reliés à la commande combinée RAMW telle que :

$$RAMW = R/\overline{W} + \overline{E}$$

Avec pour E = 1 RAMW = $R/\overline{W}$
pour E = 0 RAMW = 1.

Cela signifie que pour la phase active du 6809 E, les 4416 sont commandées directement en lecture-écriture par le microprocesseur, alors que pour la phase inactive, ces mémoires sont en lecture automatique (REFRESH).

Les deux entrées d'interruption utilisées sont IRQN et FIRQN. La première sert à gérer le clignotement du curseur (sortie du TIMER 6846) et le fonctionnement du clavier (demande provenant du 6850). L'aiguillage de l'interruption est réalisé par logiciel. La deuxième assure la gestion du crayon optique ou d'un éventuel "code barre" (demande provenant du 6821).

Lorsqu'après la mise sous tension ou après une action manuelle sur la commande de RESET, le TO9 a déroulé son menu principal, le 6809 E ayant reçu par logiciel l'instruction SYNC, celui-ci cesse toute activité et se place en attente de demande d'interruption clavier. Tous ses bus sont en haute impédance. Les buffers I21 et I22 sont, par principe, déconnectés; seuls travaillent les bus RAMA et RAMB et le bus d'adresses multiplexées, en relation avec le gate array principal, pour le rafraîchissement des mémoires 4416 et de l'écran.

3. Système de mémorisation

Les ROMS

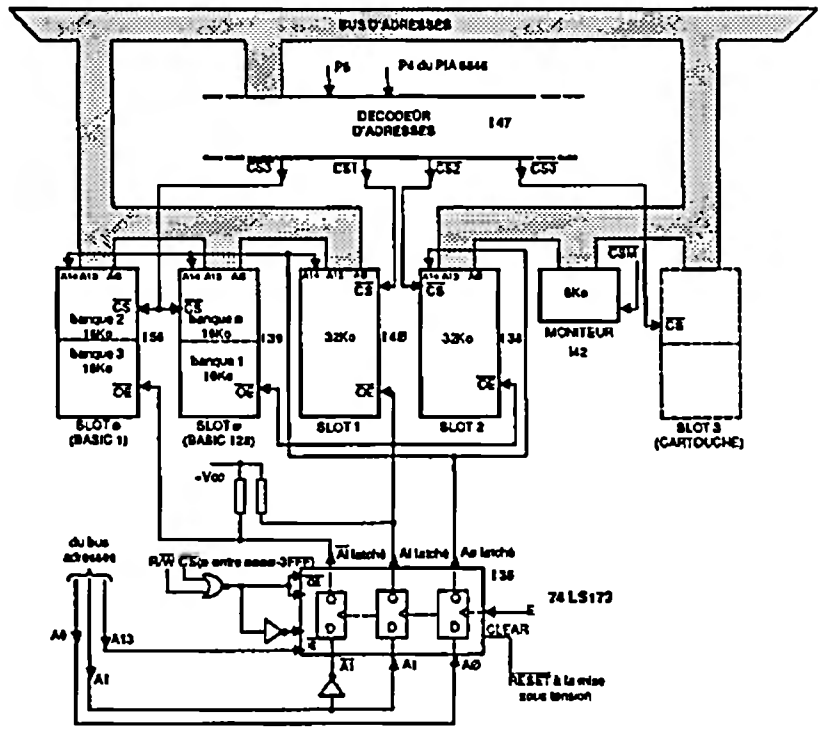


Figure 4. Gestion des ROMS dans le TO9

La figure 4 décrit l'organisation générale du fonctionnement.

Le TO9 comprend quatre EPROMS 27256 (32 Ko de logiciel intégré) et une EPROM 2764 (8 Ko de moniteur). Il peut, de plus, supporter une cartouche MEMO 7. L'espace mémoire implicitement réservé aux logiciels étant de 16 Ko, afin de pouvoir y loger les 4 × 32 Ko + MEMO 7, le système est organisé en slots et banques commutables de 16 Ko.

Chaque EPROM appartient à un slot défini par la commande de CHIP SELECT: CSN. On peut voir aussi que I-56 et I-39 (BASIC 1 + DIVERS et BASIC 128 + EXTRAMON) appartiennent au même slot 0. I-40 représente le

slot 1; I-38 représente le slot 2. Le slot 1 ou le slot 2 contiennent indifféremment le progiciel PARAGRAPHE ou FICHES ET DOSSIERS. Enfin, la cartouche extérieure appartient au slot 3.

Commutation des slots

Elle se définit par programmation, selon l'état des bits P5 et P4 en provenance du port du 6846. De par le décodage d'adresses I-47, les commandes de CHIP SELECT sont alors orientées en conséquence vers les EPROMS. Le tableau suivant met en évidence les différentes combinaisons possibles :

Adressage	P5	P4	Slot Actif
0000-3FFF	0	0	0
- -	0	1	1
- -	1	0	2
- -	1	1	3
4000-FFFF	X	X	aucun

Commutations des banques 16 Ko

Deux commandes interviennent au niveau des EPROMS: le bit A14 définissant le partage de 16 Ko partie basse ou 16 Ko partie haute; l'entrée "OUTPUT ENABLE OEN" sélectionnant le circuit en sortie.

De par le montage, la commutation est alors définie en mémorisant les deux bits d'adresse A1 et A0 via le registre 74 LS 173 I- 35. Les différents cas de figure réalisable sont représentés par le tableau suivant:

Adressage	A1 latché	A0 latché	Banque Active
0000-3FFF	0	0	0
- -	0	1	1
- -	1	0	2
- -	1	1	3
4000-FFFF	X	X	aucune

A1 et A0 sont latchés par programmation. Ainsi, il suffit de demander une écriture du 6809 E dans la zone d'adressage 0000-1FFF. Cette action implique :

R/WN = 0 CSN = 0 A13 = 0

et conformément au câblage, a pour conséquence de valider, en entrée, I-35 qui enregistre alors au coup d'horloge E les valeurs de A1 et A0 présentes à cet

instant. Conjointement, les sorties latchées sont déconnectées par OEN, ce qui représente un état haut (résistances de PULL UP), évitant un conflit au niveau des EPROMS.

A la mise sous tension, les bascules de I-35 sont réinitialisées à 0 (RESET). On notera que la zone d'adressage 2000 à 3FFF est réservée à un autre système utilisant un registre extérieur.

Fonctionnement d'ensemble: espace mémoire 0000-1FFF

Bits de PIA		Adresses latchées		Désignation		
P5	P4	A1	A0	slot	banque	logiciel
0	0	0	0	0	0	BASIC 128
0	0	0	1	0	1	extramon
0	0	1	0	0	2	BASIC 1
0	0	1	1	0	3	DOS iconique
0	1	0	X	1	X	Paragraphe ou
1	0	0	X	2	X	Fiches et Dossiers
1	1	0	0	3	X	cartouche

On peut voir que le système est prévu pour supporter quatre slots de quatre banques chacune.

Routines de commutation

Le moniteur comporte au point d'entrée standard EC03 une routine COMSLOT permettant d'appeler une routine d'une banque quelconque d'un slot quelconque (voir page 222). Les commutations effectuées s'opèrent alors à partir d'un mot binaire désigné par l'utilisateur selon la forme :

Les RAMS

Technologiquement, la mémoire vive du TO9 est constituée par 16 boîtiers intégrés du type 4416. Ces circuits ont une capacité de 16 Ko × 4 bits. Sachant que le système travaille en données de 8 bits, deux 4416 sont associées pour chaque plan mémoire de 16 Ko. On obtient ainsi huit couples de RAM, avec, pour chaque couple, un boîtier spécialisé pour les quatre bits de poids faible et un boîtier spécialisé pour les quatre bits de poids fort du bus de données.

Fonctionnement d'une 4416

Les 4416 sont des mémoires qui permettent de stocker, sous forme de matrices de $2^8 = 256$ lignes et de $2^6 = 64$ colonnes: $256 \times 64 = 16\,384$ groupes de 4 bits. L'adressage d'une telle matrice nécessite donc 14 bits. En fait, seuls 8 bits d'adresses (A0 à A1) permettent la gestion de la mémoire. Ils sont multiplexés et dirigés, soit vers le bloc d'adresses lignes, quand le signal de validation lignes RASN passe à zéro, soit vers le bloc d'adresses colonnes, quand le signal de validation colonnes CASN passe à zéro.

Les 4416 sont des mémoires du type MOS dynamique nécessitant un rafraîchissement (REFRESH) de cycle ≤ 2 ms. En groupe de 4 bits, ce rafraîchissement se fait par adressages successifs des 256 lignes. A chaque ligne adressée, si RASN est actif (état 0), les 64 cellules à transistors, placées aux intersections de cette ligne avec les 64 colonnes, sont simultanément rafraîchies. Le tableau suivant précise la forme de l'adressage:

a0	X
a1	a8
a2	a9
a3	a10
a4	a11
a5	a12
a6	a13
a7	X
Adressage lignes validé par RASN et rafraîchissement d'une ligne	Adressage colonnes validé par CASN

Les 4416, enfin, ne sont pas sélectées (bus de données déconnecté) lorsque le signal de CASN n'est pas actif. Ce dernier remplace avantageusement une commande de CHIP SELECT (action conjointe de GN).

Principe fondamental

La figure 5 schématise l'organisation générale du système. On y distingue les huit plans mémoire, partie A, partie B et banques de 0 à 5. L'adressage des RAMS ne provient pas en direct du 6809 mais passe à travers le gate array (circuit I-25) par un double multiplexage.

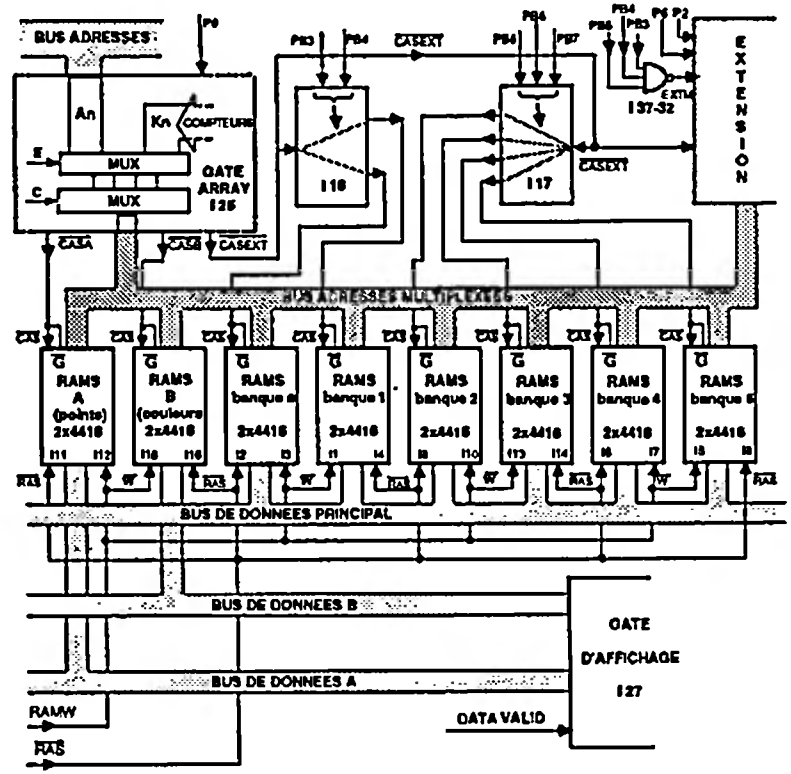


Figure 5. Système de mémorisation RAMS dans le TO9

Le premier jeu de multiplexeurs permet de commuter, au rythme du signal d'horloge E, tantôt le bus d'adresses du CPU (phase active du 6809 E, E = 1), tantôt un ensemble de compteurs intégrés (phase non active du 6809 E, E = 0). Ce processus est une technique de DMA permettant à chaque cycle d'horloge le rafraîchissement d'une ligne de RAM dynamique, à savoir le rafraîchissement total des boîtiers lorsque les compteurs ont effectué un cycle de 256 pas. Il permet, d'une tout autre manière, "le rafraîchissement d'écran" par l'intermédiaire d'une partie des RAMA et RAMB (cf. gestion d'écran).

Le deuxième jeu de multiplexeur assure la commutation poids faible et poids fort de l'adressage lignes et colonnes conformément aux spécifications des 4416; ce que confirme le tableau suivant.

Forme de l'adressage complet (double multiplexage)

A0	X	T0	X
A1	A8	T1	T8
A2	A9	T2	T9
A3	A10	T3	T10
A4	A11	T4	T11
A5	A12	T5	T12
A6	A13	T6	0
A7	X	T7	X
poids faible	poids fort	poids faible	poids fort

adressage 6809
E = 1 lecture écriture du CPU

adressage compteur
E = 0 refresh écran

On remarquera que le système de comptage est limité au bit de poids 12. Cela est justifié par le fait que les 2 × 8 Ko de RAM écran sont répartis dans la moitié inférieure des plans mémoire A et B (cf. organisation mémoire).

Organisation mémoire vive du TO9

Les huit couples mémoire organisés **physiquement** sont structurés **logiquement** en dix parties. Ainsi :

La RAMA comprend:
– de 4000 à 5FFF, 8 Ko de mémoire écran appelée RAMA ou RAM points (dénomination TO7, cf. modes d'affichage),
– de 6000 à 7FFF, 8 Ko de mémoire datas ou utilisateur dont de 6000 à 60FF les registres du moniteur et de 6100 à 62FF une partie réservée au système, à savoir: les diverses pages 0 de BASIC ou autres qui représentent des applications.

La RAMB comprend :
– de 4000 à 5FFF, 8 Ko de mémoire écran appelée RAMB ou RAM couleurs (dénomination TO7),
– de 8000 à 9FFF, 8 Ko de mémoire datas ou utilisateur.

Les banques 0, 1, 2, 3, 4 et 5 constituent chacune 16 Ko de mémoire datas ou utilisateur dans la zone d'adressage A000-DFFF.

Sélections

Les sélections se font à partir de l'entrée dynamique CAS, en relation avec les signaux CASAN, CASBN et CASEXTN dépendant de décodages d'adresses (cf. fonctionnement du gate array principal) et des bits de PIA dont P0 (bit de forme) du 6846 et PB3, PB4, PB5, PB6, PB7 du 6821 (technique de bank switching). L'extension de 4 × 16 Ko utilisée en disque virtuel (commandé uniquement par le contrôleur de lecteur), utilise les bits P2 et P6 du 6846 ainsi que les bits PB5, PB4, PB3 du 6821.

Les tableaux suivants mettent en évidence les commutations requises pour les différents types de fonctionnement selon la valeur de E (phase active ou non active):

Commutations des parties de RAM A et B pour E = 1 phase active du 6809 E

Zones d'adressage	P0	CASAN	CASBN	CASEXTN	Sélection
4000 - 5FFF	0	inactif	actif	inactif	couleur
4000 - 5FFF	1	actif	inactif	inactif	points
6000 - 7FFF	X	actif	inactif	inactif	Adatas
8000 - 9FFF	X	inactif	actif	inactif	Bdatas
A000 - DFFF	X	inactif	inactif	actif	banques

Commutation des parties de RAM A et B pour E = 0 phase non active du 6809 E:

Zone d'adressage	P0	CASAN	CASBN	CASEXTN	Sélection
4000 - 5FFF	X	actif	actif	inactif	coul.et points

Pour ce type de fonctionnement, l'adressage est implicite. On notera la double prise en compte sur 16 bits des datas, via le bus de données A et le bus de données B, par le gate d'affichage I-27 (rôle de DATA VALID).

Commutation des banques:

CASEXTN	P6	P2	PB7	PB6	PB5	PB4	PB3	Sélection
inactif	X	X	X	X	X	X	X	aucune
actif	X	X	1	1	1	1	0	banque 0
actif	X	X	1	1	1	0	1	banque 1
actif	X	X	0	0	0	1	1	banque 2
actif	X	X	1	0	0	1	1	banque 3
actif	X	X	0	1	0	1	1	banque 4
actif	X	X	1	1	0	1	1	banque 5
actif	0	0	1	1	1	1	1	RAM disque 1
actif	1	0	1	1	1	1	1	RAM disque 2
actif	0	1	1	1	1	1	1	RAM disque 3
actif	1	1	1	1	1	1	1	RAM disque 4

Lecture-écriture des RAMS

Les 4416 sont en écriture pour l'entrée W à 0 et en lecture pour W à 1. Chaque entrée W est reliée à la commande RAMW. Il en résulte une lecture automatique des RAMS pendant la phase non active du CPU(cf. le 6809 E et ses bus). Ce procédé est bien naturel puisqu'on le sait, il faut réaliser à ce moment précis le rafraîchissement d'écran. Ainsi, lorsque E = 0 le gate array d'affichage I-27 va pouvoir lire automatiquement et simultanément le contenu d'une adresse RAM point et RAM couleur (rôle des bus A et B).

Routines de commutation de banques

L'extension mémoire de 64 Ko n'étant pas considérée, dans les applications soft, comme de la mémoire de programme mais comme un disque virtuel, le sous-programme suivant ne tient donc pas compte de cette extension à laquelle on accède directement par le contrôleur de disque.

Pour des raisons **impératives** de protection de sélection simultanée de plusieurs banques RAM, le processus suivant a été établi :

– Les 5 bits de données du PIA 6821 qui servent à la sélection sont toujours à zéro (dans le registre de sortie ORB).
– Pour commuter une banque, on n'écrira jamais dans ce registre, mais on **changera la direction des bits concernés.** Cela se produira par une écriture dans le registre de direction de données (DDRB), en sachant qu'un bit en entrée génère, par technologie, un "1" et un bit en sortie génère un "0" en provenance de l'ORB.

Il est bon aussi de faire attention à certaines routines du moniteur qui peuvent modifier temporairement le contenu du PIA et restaurer tout à "0" à la fin.

Sous-programme standard:
Entrée: (registre A 6809) = numéro de la banque 0 à 5.

```
COMMUT  EQU   *
        PSHS  D,X,U
        LDU   #$E7C0      U pointe sur les PIA
        LDB   11,U        Lecture du registre CRB en E7CB
        ANDB  #$FB        Passage du PIA en mode direction
        STB   11,U        "     "     "
        LDX   #TAB        X pointe sur l'adresse de la table
                          des valeurs à mettre dans le PIA
        LDA   A,X         Lecture de la valeur de la table
                          en fonction du n° de banque
        STA   9,U         Modifications des directions
                          écriture dans le DDRB du PIA en E7C9
        ORB   #$04        Passage du PIA en mode données
        STB   11,U        (retour aux conditions départ)

        PULS  D,X,U,PC    Retour

TAB     EQU   *
        FCB   $0F,$17,$E7,$67,$A7,$27
```

Le tableau ci-après précise l'action du conteou de TAB en relation avec le contenu de A.

(A)	(TAB) pointé	résultante valeur dans le DDRB	configuration du PIA PB7	PB6	PB5	PB4	PB3	Sél.banq.
0	0F	0 0 0 0 1 1 1 1	1	1	1	1	0	0
1	17	0 0 0 1 0 1 1 1	1	1	1	0	1	1
2	E7	1 1 1 0 0 1 1 1	0	0	0	1	1	2
3	67	0 1 1 0 0 1 1 1	1	0	0	1	1	3
4	A7	1 0 1 0 0 1 1 1	0	1	0	1	1	4
5	27	0 0 1 0 0 1 1 1	1	1	0	1	1	5

4. La gestion du système

Les décodages d'adresses

Le décodage d'adresses du TO9 est effectué par un circuit intégré quarante broches spécialisé. C'est un gate array TAHC06 de chez TEXAS regroupant un réseau câblé de portes et de décodeurs TTL type 74 LS 156, 74 LS 138 et 139.

Ce circuit a quinze entrées concernant le bus d'adresses de A1 à A15, et les deux bits de PIA en provenance du 6846 pour la commutation des EPROMS. Seize sorties sont distribuées pour effectuer les sélections suivantes:

- de E7DE à E7DF, pour la validation du circuit ACIA 6850 (gestion clavier)
- de E7DC à E7DD, pour la validation du registre mode d'affichage du gate array I-27,
- de E7DA à E7DB, pour valider les registres du circuit de palette I-28,
- en E7D8, pour la validation du registre de commande double densité et choix des lecteurs I-50,
- de E7D0 à E7D3, pour la validation des registres du contrôleur de lecteur,
- de E000 à E7AF et de E800 à FFFF, pour valider le moniteur,
- de E7C0 à E7CF, pour la validation du 6846, du 6821 système et du 6821 contrôleur de jeux externes,
- de 4000 à 9FFF, pour déconnecter le buffer du bus de données principal, lorsque le 6809 E travaille en RAMA ou en RAMB (commande VP du 74 LS 245),
- de 4000 à DFFF, pour la validation des plans RAM, selon trois signaux:

CSAN = $\overline{CSA}$ actif à "0" de 4000 à 7FFF

CSBN = $\overline{CSB}$ actif à "0" de 8000 à 9FFF

CSEXTN = $\overline{CSEXT}$ actif à "0" de A000 à DFFF

- de 0000 à 3FFF, pour déterminer l'espace alloué à la commutation des banques EPROMS (commande reliée au 74LS173 I-35) et générer:

$\overline{CS0}$, $\overline{CS1}$, $\overline{CS2}$, $\overline{CS3}$.

Une sortie particulière: CSEN = $\overline{CSE}$ active à "0" pour l'espace EXXX, (soit de E000 à EFFF), est destinée aux différentes extensions ou contrôleurs de communication. Elle permet, entre autres, d'effectuer un prédécodage d'adresse dans le cas où l'utilisateur désire se fabriquer sa propre extension. Par un montage approprié, il devra alors impérativement loger cette dernière dans l'espace mémoire E7B0 à E7BF ou E7E0 à E7E3.

La génération des fonctions

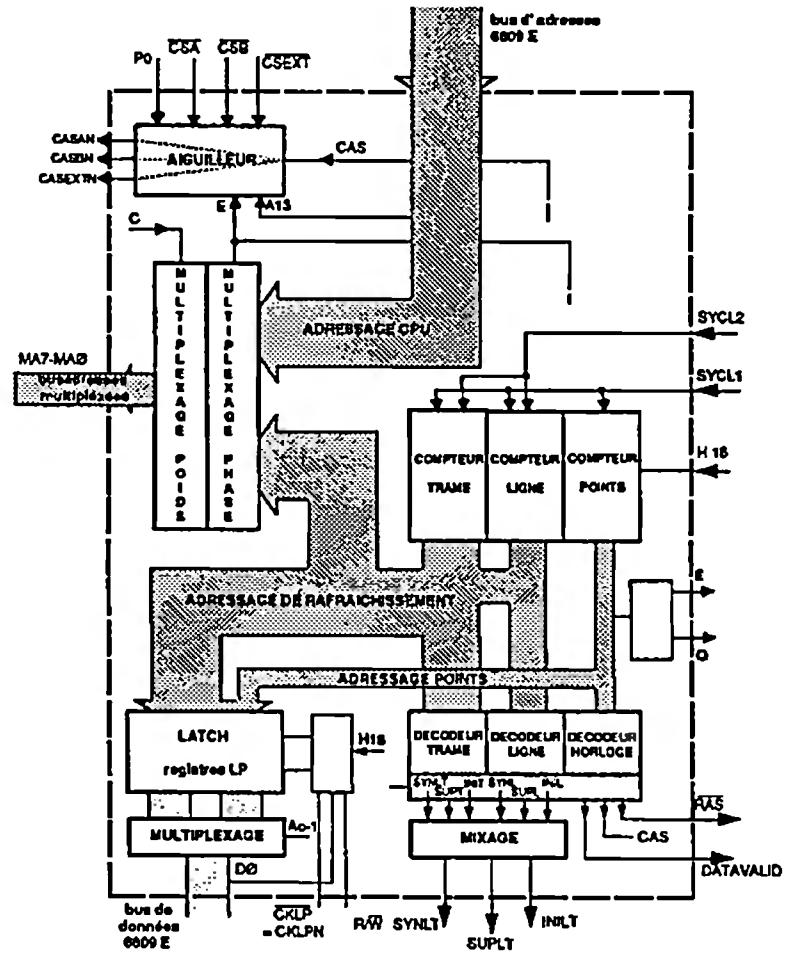


Figure 6. Synoptique du gate array principal dans le TO9

Le gate array principal ou gate array système

Le gate array principal, type EFGJ03 (circuit I-25) de chez EFCIS, est un circuit à réseau prédiffusé servant de générateur de fonctions. Il a quatre rôles fondamentaux:

– générer des signaux d'horloge et de comptage point,
– générer des signaux d'interfaçage vidéo, en relation avec des compteurs lignes et trames,
– gérer les mémoires vives dynamiques 4416,
– récupérer et stocker le contenu des compteurs points, lignes et trames pour le traitement du crayon optique

Description fonctionnelle

Signaux d'horloge et compteur points

H16 est une entrée d'horloge 16 MHz (horloge mère externe). Elle est divisée, de proche en proche, par un compteur à quatre étages, créant les fréquences points: H4, H2 ainsi que E et Q, en quadrature, à la fréquence de 1 MHz pour le microprocesseur.

Un décodeur d'horloge fournit trois signaux tels que:
RASN, CAS, à la fréquence de 2 MHz ainsi que DATAVALID à la fréquence de 1 MHz.

L'entrée SYCL1 permet de réinitialiser l'ensemble du système.

Compteur lignes et génération des signaux vidéo

Compteur lignes: E, de fréquence 1 MHz, pilote un compteur 6 bits dont les sorties sont successivement, TL0, TL1, TL2, TL3, TL4, TL5. Le cycle de comptage dure 64 µs, durée normalisée d'une ligne de balayage TV.

L'entrée SYCL2 réinitialise le compteur.

Décodeur ligne: En relation avec les sorties du compteur, il génère les signaux de synchro, de suppression et d'inhibition pour le téléviseur.

INIL est le signal d'inhibition ligne permettant de définir la fenêtre de visualisation sur l'écran pour une largeur correspondant à un balayage dont la durée est 40 µs.

SUPL est le signal de suppression ligne. Il est destiné à mettre au niveau du noir les signaux RVB, durant le retour du spot en ligne.

SYNL est le signal de synchronisation ligne.

Compteur trame et génération des signaux vidéo

Compteur trame de 3 + 11 = 14 bits dont les trois premiers étages sont, en réalité, les trois premiers étages du compteur lignes. Les sorties sont repérées TL0, TL1, TL2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13. L'horloge de commande est une combinaison logique de E et INIL, dont une inhibition de 24 µs toutes les 64 µs. Autrement dit, le compteur ne fonctionne que pendant le balayage de la fenêtre de visualisation. Ce compteur est réinitialisé par un signal interne à chaque trame TV. Il peut être réinitialisé comme le compteur lignes par l'entrée SYCL2.

Décodeur trame: En relation avec les sorties du compteur, il génère les signaux de synchro, de suppression et d'inhibition trame. En standard SECAM, le signal de synchro est distribué toutes les 312 lignes, soit toutes les 312 × 64 = 19 968 µs (durée d'une trame environ 20 ms).

INIT est le signal d'inhibition trame permettant de définir la fenêtre de visualisation sur l'écran pour une hauteur dont la durée correspond à un balayage de 200 lignes.

SUPT est le signal de suppression trame. Il est destiné à mettre au niveau du noir les sorties RVB pendant le retour trame.

SYNT est le signal de synchronisation trame.

Mixage des signaux lignes-trames

Le gate array dispose de trois sorties de signaux lignes et trames mélangés tels que:

– INILT signal d'inhibition ligne et trame:

INILT = INIL + INIT

– SYNLT signal de synchro ligne et trame:

SYNLT = SYNL + SYNT

– SUPLT signal de blanking:

SUPLT = SUPL + SUPT

Gestion des mémoires vives

Le gate array comprend un double circuit de multiplexeurs recevant le bus d'adresses CPU et le bus compteur ligne et trame. Le double multiplexage est piloté par E et un signal interne C, dérivé du signal CAS. Il permet de distribuer sur 9 bits une succession d'adresses basses et hautes de provenance CPU ou compteur selon le diagramme suivant:

A0	A7	TL0	T7
A1	A8	TL1	T8
A2	A9	TL2	T9
A3	A10	T3	T10
A4	A11	T4	T11
A5	A12	T5	T12
A6	A13 + CSB	T6	0
A7	A14	T7	0
A8	A15	0	0
	E = 1		E = 0

On notera la relation logique A13 + CSB dont le rôle est expliqué ci-après. Dans le TO9 le neuvième bit n'est pas exploité.

Un circuit "aiguilleur de CAS" distribue, en relation avec la combinaison logique de A13, CSAN, CSBN, CSEXTN, le bit FORME (P0) et la validation de E, les signaux CASAN, CASBN ainsi que CASEXTN selon le tableau suivant:

E	A13	FORME (P0)	CSAN	CSBN	CSEXTN	CASAN	CASBN	CASEXTN
0	X	X	X	X	X	CASN	CASN	1
1	0	0	0	1	1	1	CASN	1
1	0	1	0	1	1	CASN	1	1
1	1	X	0	1	1	CASN	1	1
1	X	X	1	0	1	1	CASN	1
1	X	X	1	1	0	1	1	CASN

De par le décodage d'adresses, selon les trois champs d'adresses suivants, nous savons par ailleurs que:

De 4000-7FFF	CSAN = 0	CSBN = 1	CSEXTN = 1
De 8000-9FFF	CSAN = 1	CSBN = 0	CSEXTN = 1
De A000-DFFF	CSAN = 1	CSBN = 1	CSEXTN = 0

Ce qui permet de construire le tableau général de la commutation des plans mémoires pour E = 1 (cf. organisation mémoire vive du TO9, page 42).

Champ mémoire	A13	P0	$\overline{CASA}$	$\overline{CASB}$	$\overline{CASEXT}$	A13+CSB	Type de RAM
4000-5FFF	0	0	1	CASN	1	0	RAMB partie basse (écran couleur)
4000-5FFF	0	1	CASN	1	1	0	RAMA partie basse (écran forme)
6000-7FFF	1	X	CASN	1	1	1	RAMA partie haute (datas)
8000-9FFF	0	X	1	CASN	1	1	RAMB partie haute (datas)
A000-DFFF	X	X	1	1	CASN	X	banques

On met ainsi en évidence la sélection des plans mémoires A (écran forme) et B (écran couleur) par le bit de forme P0; de même, la sélection des plans mémoires A (datas) et B (datas) par l'action de A7 + CSB envoyé sur le bus d'adresses mutiplexé.

Traitement du crayon optique

Une série de *flip-flop*, dénommée latch ou registre crayon optique, située aux adresses E7E4 E7E5 E7E6 E7E7, réalise la prise en compte des états présents des compteurs, au moment d'une visée sur l'écran par le crayon optique. Dès que l'utilisateur pointe le crayon sur la fenêtre de visualisation, une impulsion (niveau bas) de 700 ns apparaît sur l'entrée CKLPN. Cette impulsion synchronisée sur le premier front descendant de H16 est appliquée en commande d'horloge des *flip-flop*. Ces derniers reçoivent sur leur entrée l'état des compteurs points, lignes et trame afin de les stocker.

Pour R/WN = 1, les informations mémorisées dans les *flip-flop* sont multiplexées par les bits A0 et A1 du bus d'adresses et envoyées sur le bus de datas D0 à D7. Quand le circuit n'est pas sélectionné, ces sorties sont normalement en tristate. Pour R/WN = 0, D0 peut passer en entrée.

Pour lire les informations, le 6809 doit faire appel à une routine particulière. Il doit envoyer, en écriture, un front montant sur D0 à l'adresse E7E4. Le gate array autorise alors le passage du signal issu de CKLPN en rendant disponibles les données en sortie. Le 6809 peut lire les données à travers l'état de A1 et A0 en envoyant les adresses E7E4 à E7E7.

Les informations mémorisées permettent la lecture de la position du pointage sur l'écran en donnant la précision du point, dans une totalité de 8000 (comptage sur 16 bits de H4 à T12). La prise en compte de INILN et INITN permet de vérifier si le crayon optique a été réellement pointé sur la fenêtre de visualisation (cf. l'étude du fonctionnement du crayon optique).

5. Le système de visualisation

Généralités

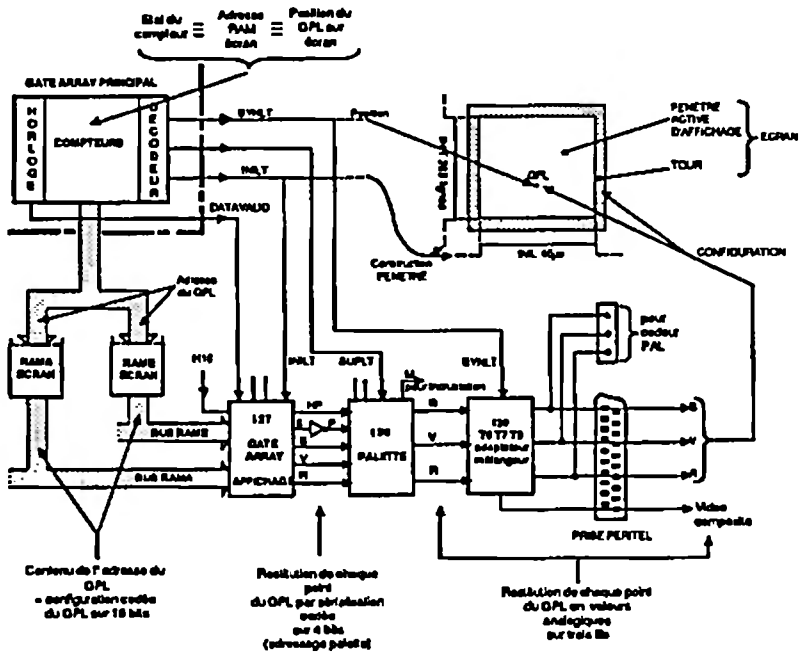


Figure 7. Synoptique du système de visualisation

La figure 7 donne un aperçu général de la gestion d'écran TV. Il est bon de décomposer le fonctionnement en deux parties.

Construction globale de l'écran

L'écran doit être divisé en deux zones distinctes :
– La fenêtre active d'affichage ou de visualisation, dont la dimension en largeur correspond à un temps de balayage de 40 µs et la dimension en hauteur à 200 lignes de 64 µs. La fenêtre est destinée à recevoir caractères et graphismes.

– Le tour ou cadre ne recevant ni caractère ni graphisme mais susceptible de changer de teinte.

La définition géométrique de la fenêtre et du cadre est réalisée à partir du signal INILT, distribué par le gate array principal, et venant agir sur un aiguilleur contenu dans le gate array d'affichage I-27. Ce dernier détermine, entre autres, les couleurs du tour.

La définition de l'écran est faite à partir des signaux SYNLT et SUPLT (synchronisme ligne, trame et blanking) pour assurer un balayage non interligné de 312 lignes de durée normalisée à 64 µs.

Construction de la fenêtre active

La fenêtre de visualisation ou de travail est réalisée en synchronisme avec le balayage du spot par l'association de segments en ligne et en colonne. Ces segments sont appelés GPL. Quarante GPL constituent une ligne. La fenêtre est donc construite à partir de 40 × 200 lignes = 8 000 GPL.

Un GPL est conçu par une association de points (4, 8, 16 selon le mode d'affichage). Chaque point, représentatif d'une couleur, est déterminé à partir d'une ou plusieurs informations stockées en mémoire écran. En fait, le contenu 16 bits d'une adresse de mémoire écran définit la configuration en couleur des points d'un GPL. Chaque GPL est donc associé à une adresse qui représente sa position sur la fenêtre. Pour gérer 8 000 GPL, il faut 8 000 adresses dans le champ 4000 à 5F3F.

Pendant chaque phase non active du 6809 E, nous savons que chaque nouvelle adresse pointée est incrémentée de 1 (cf. fonctionnement du gate array principal, page 48); ce qui signifie que toutes les µs, un nouveau GPL est défini en parfait synchronisme avec le balayage, puisque SYNLT est issu des compteurs délivrant les adresses. Ce processus est appelé rafraîchissement d'écran ou REFRESH.

Le contenu de la RAM écran RAMA et RAMB représente donc la configuration codée sur 16 bits des GPL. Ce contenu est dirigé par les bus RAMA et RAMB vers le circuit I-27 (gate array d'affichage). Ce dernier, selon le mode d'affichage programmé, restitue des informations de couleurs sérialisées sur le bus "couleur" de 4 bits. Trois bits représentent une couleur primaire R (rouge), V (vert), B (bleu) correspondant à la configuration de base (couleur du TO7/70). Le quatrième bit ressort l'information de saturation S transformée, après inversion, en information de pastel P. L'association de ces quatre bits définit un adressage ou un numéro de couleur pour le circuit de palette.

Les quatre bits activent, en relation avec une horloge PIXEL (HP), le circuit I-28 de palette dont le rôle principal est de convertir en niveaux analogiques le codage binaire de chaque point d'un GPL. Optionnellement, il permet de

reconfigurer les couleurs de base, grâce à un plan mémoire programmable, pouvant traiter 4 096 cas de figures.

Trois sorties analogiques RVB attaquent la prise péritel aux normes SCART à travers un circuit d'adaptation I-30 et un mélangeur recevant SYNLT pour reconstituer une vidéo composite. Les trois signaux sont dérivés vers un adaptateur pour un éventuel codeur PAL.

Configuration de base

A la mise sous tension, le circuit de palette I-28 est programmé pour restituer, en relation avec l'adressage par numéros de couleur provenant du gate array I-27, les couleurs fondamentales du TO7/70, selon le tableau suivant :

Adressage en sortie gate d'affichage				Adressage en entrée palette				Numéro de couleur	Sélection de couleur du TO7/70
S	B	V	R	P	B	V	R		
1	0	0	0	0	0	0	0	0	noir
1	0	0	1	0	0	0	1	1	rouge
1	0	1	0	0	0	1	0	2	vert
1	0	1	1	0	0	1	1	3	jaune
1	1	0	0	0	1	0	0	4	bleu
1	1	0	1	0	1	0	1	5	magenta
1	1	1	0	0	1	1	0	6	cyan
1	1	1	1	0	1	1	1	7	blanc
0	0	0	0	1	0	0	0	8	gris
0	0	0	1	1	0	0	1	9	rose
0	0	1	0	1	0	1	0	10	vert clair
0	0	1	1	1	0	1	1	11	sable
0	1	0	0	1	1	0	0	12	bleu clair
0	1	0	1	1	1	0	1	13	parme
0	1	1	0	1	1	1	0	14	bleu ciel
0	1	1	1	1	1	1	1	15	orange

Gate array d'affichage et modes d'affichage

Rôle du circuit I-27

Le gate array EFGG06 de chez EFCIS a été conçu pour récupérer les 16 bits de données en provenance des RAMS écran afin de pouvoir, en les sérialisant pendant une micro-seconde sur un bus de quatre fils (R,V, B, S), construire, via l'interface vidéo, un GPL selon un mode d'affichage bien particulier.

Ce circuit comporte des registres programmables par le 6809 E, pour définir un des huit modes d'affichage tels que :

Le mode TO7/70 ou 40 colonnes
Le mode bit-map 4 couleurs
Le mode 80 colonnes
Le mode bit-map 16 couleurs
Le mode page 1
Le mode page 2
Le mode surimpression 1 (2 pages)
Le mode surimpression 3 (4 pages).

Description du circuit

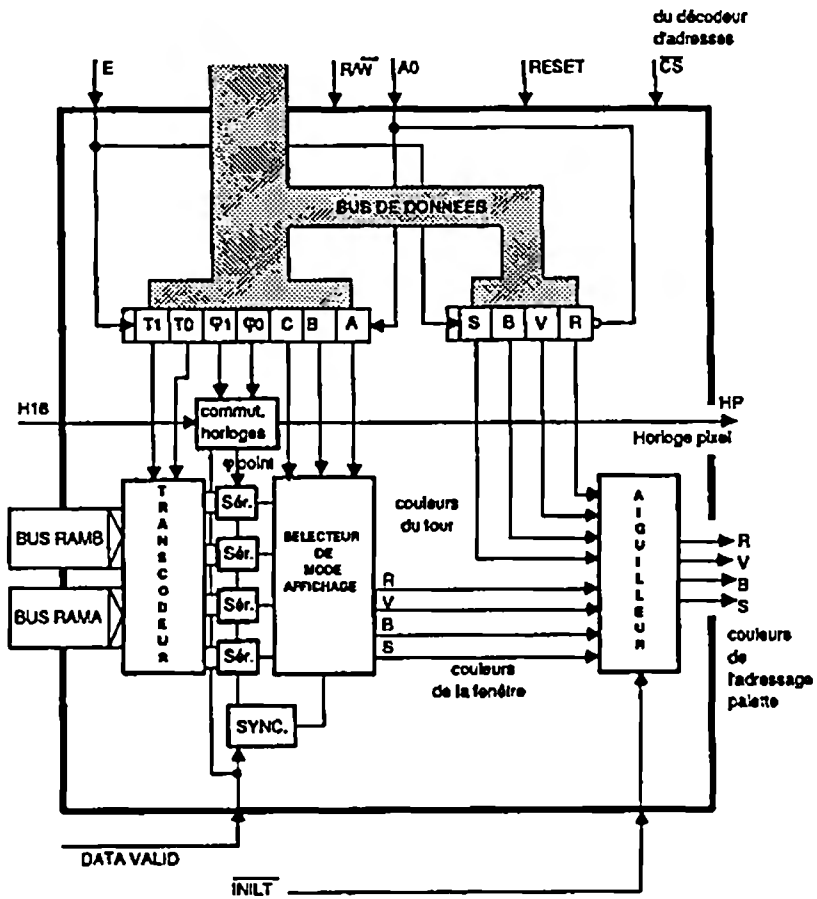


Figure 8. Synoptique du gate array d'affichage EFGG06 (circuit I-27)

Le gate array I-27 est constitué par :

– Un transcodeur en relation avec le bus RAMA et le bus RAMB. Ce transcodeur intervient uniquement pour le mode TO7/70 et le mode bit-map 16 couleurs.

– Quatre registres sérialisateurs de quatre bits chacun recevant les données transcodées et les ressortant en série. Selon le mode programmation, ces registres peuvent travailler séparément ou en association.

– Un circuit de commutation assure les différentes combinaisons. Un commutateur d'horloge sélectionne la fréquence de sérialisation Φ point de 4 MHz, 8 MHz ou 16 MHz. Il génère une fréquence dépendante de Φ point. Cette fréquence appelée HP (horloge pixel) est destinée à la prise en compte des informations du bus couleur R, V, B, S pour les registres de palette.

– L'ensemble de ces circuits est géré par un registre programmable à l'adresse E7DC. Il est accessible uniquement en écriture selon le modèle sur huit bits :

X T1 T0 Φ1 Φ0 C B A

avec choix du transcodage :

T1	T0	
0	0	Mode TO7/70
0	1	Mode sans transcodage
1	1	Mode bit-map 16 couleurs

avec choix de la fréquence d'horloge :

Φ1	Φ0	
0	0	8 MHz
0	1	16 MHz
1	1	4 MHz

avec choix du mode de fonctionnement :

C	B	A	
0	0	0	Mode TO7/70
0	0	1	Mode bit-map 4
0	1	0	Mode 80 colonnes
0	1	1	Mode bit-map 16
1	0	0	Mode page 1
1	0	1	Mode page 2
1	1	0	Mode surimpression 1
1	1	1	Mode surimpression 3

– Un autre registre, programmable uniquement en écriture en E7DD, permet d'enregistrer les couleurs du tour selon le modèle sur huit bits :

X X X X S B V R

avec S bit de saturation
B bit de bleu
V bit de vert
R bit de rouge
dans la configuration de base.

Ce registre est parfaitement indépendant du premier et ne concerne pas les modes d'affichage.

Le circuit I-27 est réinitialisé à chaque mise sous tension ou lorsque l'on appuie sur le bouton correspondant. La commande est commune avec celle du 6809 E. Il en résulte une programmation des registres à 0, d'où le mode d'affichage implicite TO7/70 et la couleur du cadre en gris.

Les différents modes d'affichage

Dans l'hypothèse où pour chacun des huit points d'un GPL on cherchait à obtenir une couleur différente, il faudrait prévoir un format de RAM écran de :

8 (points) × 4 (bits de couleurs) soit 32 bits.

La mémoire écran travaillant uniquement sur un format seize bits, seuls des compromis sont envisageables quant aux nombres de couleurs exploitables et à la finesse des pixels; ce fait est illustré par les quatre modes d'affichage suivants:

• *Le mode TO7/70 ou 40 colonnes*

C'est le mode implicite permettant pour 16 couleurs une résolution de 320 × 200 points avec, par segment de huit points, une rigidité de deux couleurs.

Dans ce mode, le GPL est défini sur huit points. A chaque point correspond une couleur de forme ou une couleur de fond. La RAMA sert à mémoriser l'emplacement Fn de la forme ou du fond (mémoire de forme). Un bit à 0 correspond à du fond, un bit à 1 correspond à de la forme. La RAMB sert à mémoriser la couleur de fond et la couleur de forme, selon les primaires R, V, B et le bit de saturation S.

Schéma de codage en RAM

RAMA								RAMB							
F7	F6	F5	F4	F3	F2	F1	F0	S0	S1	B1	V1	R1	B0	V0	R0
								↑	forme				fond		
								fond							

L'organisation en RAMB n'est pas cohérente (pour des raisons de compatibilité avec le TO7) car le bit de saturation de fond est totalement séparé des primaires correspondantes. Dans le mode TO7/70 (CBA = 000), le gate array d'affichage va donc effectuer un transcodage (T1 T0 = 00), selon le schéma suivant :

S1 B1 V1 R1 S0 B0 V0 R0

Les bits de la RAMA vont être sérialisés à la fréquence Φ point de 8 MHz (Φ1 Φ0 = 00) et vont commuter sur les quatre sorties S, B, V, R :

la couleur de fond : S0 B0 V0 R0 pour Fn = 0

la couleur de forme : S1 B1 V1 R1 pour Fn = 1

– Exemple de routine simple permettant d'afficher une succession de couleurs différentes sur un même GPL à l'adresse $5000 :

Mode TO7/70 implicite :

```
LDA     $E7C3     Commutation du bit
ANDA    #$FE      de forme à 0
STA     $E7C3     Accès à la RAMB

LDB     #$D1      S0 S1 B1 V1 R1 B0 V0 R0
STB     $5000      1  1  0  1  0  0  0  1
                  couleur de fond N°1, couleur de forme N°2,
                  à l'adresse 5000.

ORA     #$01      Commutation du bit
STA     $E7C3     de forme à 1
                  Accès à la RAMA

LDA     #$AA      F7 F6 F5 F4 F3 F2 F1 F0
                   1  0  1  0  1  0  1  0
                  Alternance de forme et
                  de fond à l'adresse 5000.

REC     BRA       REC
```


En résumé, ce mode permet d'utiliser seize couleurs avec deux couleurs imposées par GPL, une couleur de forme, et une couleur de fond. Ce mode entraîne d'inévitables conflits de proximité dans la réalisation des graphismes (bavures ou débordement de couleurs sur un GPL).

• Le mode bit-map quatre couleurs

Ce mode est destiné à éviter les conflits de proximité, pour une résolution de 320 × 200 points, avec uniquement quatre couleurs imposées.

Le GPL est défini sur huit points. A chaque point, de gauche à droite et dans le sens décroissant des poids, correspond un bit dans la RAMA et dans la RAMB pour coder sa couleur selon la configuration de base :

RAMB	RAMA	Couleur du point
0	0	gris
0	1	rose
1	0	vert clair
1	1	sable

Schéma de codage en RAM

RAMA								RAMB							
C7	C6	C5	C4	C3	C2	C1	C0	C7	C6	C5	C4	C3	C2	C1	C0
pour chaque point présence ou absence de rouge								pour chaque point présence ou absence de vert							

Pour ce mode (CBA = 001), le gate array d'affichage n'effectue pas de transcodage (T1 T0 = 01). Les bits de la RAMA et de la RAMB vont être sérialisés à la fréquence Φ point de 8 MHz (Φ1 Φ0 = 00) chacun sur un fil, R pour la RAMA, V pour la RAMB. Les deux autres fils S et B sont à l'état 0 permanent. Ceci explique pourquoi le système génère des teintes en pastel.

– Exemple de routine simple permettant d'afficher une succession de couleurs différentes sur un même GPL à l'adresse $5000.

Mode bit-map 4 couleurs :

```
        LDA     $E7C3     Initialisation de la RAMA
        ANDA    #$FE      et de la RAMB pour un
        STA     $E7C3     écran uniforme
        CLRB              "
        JSR     EFF       "
        ORA     #$01      "
        STA     $E7C3     "
        JSR     EFF       "

        LDA     #$21      Passage en mode
        STA     $E7DC     bit-map 4

        LDB     #$CC      C7 C6 C5 C4 C3 C2 C1 C0
        STB     $5000     1  1  0  0  1  1  0  0
                          dans la RAMA

        LDA     $E7C3     Passage en RAMB
        ANDA    #$FE      "
        STA     #E7C3     "

        LDB     #$AA      C7 C6 C5 C4 C3 C2 C1 C0
        STB     $5000     1  0  1  0  1  0  1  0

REC     BRA     REC

EFF     EQU     *         Routine d'initialisation
        LDX     #$4000    "
ENC     STB     ,X+       "
        CMPX    #$5F40    "
        BNE     ENC       "
        RTS               "
```

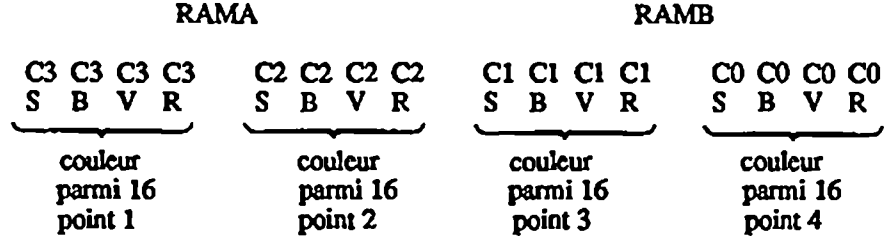
Ce programme exécuté en configuration de base détermine par principe quatre couleurs pastel : gris, rose, vert clair et sable. Afin d'obtenir un affichage plus apparent, il convient de reconfigurer la palette et de la transformer de telle sorte que le gris devienne noir, le rose devienne rouge, le vert clair devienne vert, le sable devienne jaune. Cette reconfiguration est implicitement apportée par la séquence d'échappement ($59) inhérente à la routine PUTC (cf. programmation de la palette, page 212).

• Le mode bit-map 16 couleurs

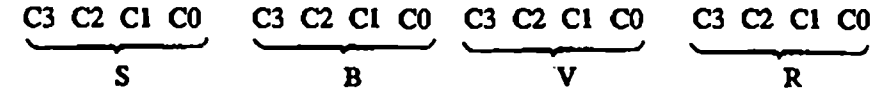
Ce mode est destiné à éviter les conflits de proximité pour une résolution de 160 × 200 points avec le choix de seize couleurs.

Le GPL est conçu sur quatre points. Pour chaque GPL, la RAMA, par ses quatre bits de poids fort, définit la couleur parmi seize du point le plus à gauche, puis, par ses quatre bits de poids faible la couleur du point immédiatement juxtaposé. La RAMB, par ses quatre bits de poids fort, définit la couleur du point suivant, enfin par ses quatre bits de poids faible, la couleur du point le plus à droite.

Schéma de codage en RAM:



Pour ce mode (CBA = 011), le gate array d'affichage effectue un transcodage (T1 T0 = 11) selon le schéma:



Quatre sérialisateurs commandés par Φ point = 4 MHz (Φ1 Φ0 = 11), spécialisés chacun en S, B, V, R vont délivrer sur les quatre fils correspondants l'information de teinte du point considéré.

– Exemple d'une routine simple permettant d'afficher une succession de couleurs différentes sur un même GPL, à l'adresse $5000:

Initialisation de la RAMA et de la RAMB
pour un écran uniforme (cf. bit-map 4 couleurs)

```
        LDA     #$7B      Passage en mode
        STA     $ E7DC    bit-map 16

        LDB     #$0C      C3 C3 C3 C3 C2 C2 C2 C2
        STB     $5000     0  0  0  0  1  1  0  0
                          dans la RAMA

        LDA     $E7C3     Passage en RAMB
        ANDA    #$FE      "
        STA     #E7C3     "

        LDB     #$A9      C1 C1 C1 C1 C0 C0 C0 C0
        STB     $5000     1  0  1  0  1  0  0  1
                          dans la RAMB
REC     BRA     REC
```

Ce programme exécuté en configuration de base, affiche quatre points gris, bleu, vert, rouge de la gauche vers la droite. On en conclut que ce mode qui n'a pas de couleurs imposées mais une résolution divisée par 2, n'entraîne pas de conflit de proximité.

• *Le mode 80 colonnes*

C'est le mode permettant la plus haute résolution 640 × 200 points avec seulement deux couleurs imposées, une couleur de forme ou une couleur de fond. Le GPL est défini sur seize points. La RAMA et la RAMB ont donc le même codage et servent à mémoriser l'emplacement de la forme ou du fond.

Schéma de codage en RAM

RAMA								RAMB							
F15	F14	F13	F12	F11	F10	F9	F8	F7	F6	F5	F4	F3	F2	F1	F0

La RAMA mémorise la structure des points les plus à gauche, la RAMB celle des points les plus à droite. En affichage de caractères, la RAMA contient les bits des points des caractères des colonnes paires, tandis que la RAMB contient ceux des colonnes impaires.

Le gate array d'affichage, pour ce mode (CBA = 010), n'effectue pas de transcodage (T1 T0 = 01). Il combine un sérialisateur unique de 16 bits, commandé par Φ point = 16 MHz (Φ1 Φ0 = 01). Les informations sérialisées ressortent sur les deux fils B et V reliés en parallèle. S et R étant figés à 0. En configuration de base, la couleur de forme exploitable est du bleu ciel et la couleur de fond du gris.

– Exemple de routine simple permettant d'afficher seize points, successivement en couleur de forme et couleur de fond sur un même GPL, à l'adresse $5000.

Initialisation de la RAMA et de la RAMB
pour un écran uniforme (cf. bit-map 4 couleurs)

```
        LDA   #$2A      Choix du mode 80 colonnes
        STA   E7DC      "

        LDB   #$AA      Alternance de forme et fond
        STB   $5000     dans la RAMB

        LDA   $E7C3     Commutation en RAMA
        ORA   #$01      "
        STA   $E7C3     "

        STB   $5000     Alternance de forme et fond dans la RAMA

REC     BRA   REC
```


A l'exécution de ce programme, on obtient huit points séparés, de couleur bleu ciel, sur un fond gris. Afin de rendre la démonstration plus représentative, il convient de reconfigurer la palette en changeant le gris en noir et le bleu ciel en rouge. Cette opération est implicitement réalisée par la routine PUTC, lorsque l'on fait appel à la séquence d'échappement ($5B).

D'une toute autre manière, le gate array d'affichage permet, selon un codage particulier en RAMA et RAMB, trois modes intrinsèquement liés.

– Le mode page 1
– Le mode page 2
– Le mode superposition deux pages.

• *Le mode page 1*

Ce mode travaille en résolution 320 × 200 avec deux couleurs imposées. Le GPL est défini sur huit points uniquement dans la RAMA en forme et en fond.

Schéma de codage en RAM:

RAMA							
F7	F6	F5	F4	F3	F2	F1	F0

Dans ce mode (CBA 100), le gate array d'affichage reprend les bits sans transcodage (T1 T0 = 01). Un sérialisateur commandé à la fréquence Φ point = 8 MHz (Φ1 Φ0 = 00) délivre les informations sur la sortie R, les trois autres fils étant à l'état 0.

• *Le mode page 2*

Parfaitement équivalent au précédent, il prend l'information GPL en RAMB.

Schéma de codage en RAM:

RAMB							
F7	F6	F5	F4	F3	F2	F1	F0

Dans ce mode (CBA = 101) non transcodé (T1 T0 = 01). Le deuxième sérialisateur commandé avec Φ point = 8 MHz (Φ1 Φ0 = 00) délivre des informations sur la sortie V, les trois autres fils étant à l'état 0.

• *Le mode superposition* deux pages ou surimpression 1, permet de reprendre simultanément les deux modes précédents. Dans ce mode (CBA = 110), le gate array utilise à la fois les deux sérialisateurs et les deux sorties R, V selon la relation:

Sortie R = R
Sortie V = V. $\overline{R}$

Cela implique une priorité de sortie pour R (V = 0 pour R = 1) et un effet de surimpression de R sur V.

– Programme de mise en évidence des trois modes:

Initialisation de la RAMA et de la RAMB
pour un écran uniforme (cf. bit-map 4 couleurs)

```
        LDA   #$24     Passage en mode page 1
        STA   $E7DC        "

        LDB   #$AA     Tracé d'un double GPL
        STB   $5000    en pointillé
        STB   $5001    dans la RAMA

        LDA   #$25     Passage en mode page 2
        STA   $E7DC        "

        LDA   $E7C3    Commutation en RAMB
        ANDA  #$FE         "
        STA   $E7C3        "

        LDB   #$FF     Tracé d'un double GPL
        STB   $50001   en continu
        STB   $50002   dans la RAMB

REPB    JSR   $E806    Boucle d'attente
        CMPB  #$41     du caractère A
        BNE   REPB         "

        LDA   #$24     Passage en mode page 1
        STA   $E7DC        "

REPA    JSR   $E806    Boucle d'attente
        CMPB  #$53     du caractère S
        BNE   REPA         "

        LDA   #$26     Passage en mode
        STA   $E7DC    superposition

REPS    JSR   $E806    Boucle d'attente
        CMPB  #$42     du caractère B
        BNE   REPS         "

        LDA   #$25     Passage en mode page 2
        STA   $E7DC        "

        JMP   REPB     Reprise
```

Lors de l'exécution en configuration de base, on affiche sur l'écran gris un double GPL en continu vert clair. En appuyant sur la touche A, on affiche un double GPL en pointillé rose. En appuyant sur la touche S, on affiche les deux GPL, le rose empiétant sur le vert clair. La touche B permet de revenir à l'état initial.

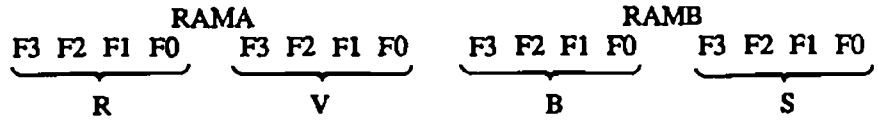
Afin d'obtenir une représentation plus apparente, il convient de reconfigurer la palette et de la transformer, de telle sorte que le gris devienne noir, le rose devienne rouge, le vert clair devienne vert saturé. Cette action est implicitement apportée par les séquences d'échappement ($48, $49, $4A ou $4B) inhérentes à la routine PUTC.

Conclusion: L'avantage de ces modes est de pouvoir offrir des pages indépendantes. Pendant l'affichage d'une page, l'autre peut être écrite et présentée ensuite par simple commutation. Le mode surimpression permet de superposer les deux pages en continuant d'écrire dans l'une ou l'autre.

• *Mode particulier*

Le mode triple surimpression ou quatre pages superposées est un mode particulier permettant d'associer des plans avec une priorité à l'affichage, pour une résolution de 160 × 200 points et cinq couleurs imposées. Le GPL est donc conçu sur quatre points. En configuration de base, chaque point de gauche à droite a sa couleur de forme définie soit en rose, soit en vert clair, soit en bleu clair, soit enfin en noir, avec une couleur de fond gris.

Schéma de codage en RAM:



Pour ce mode (CBA = 111), le gate array d'affichage n'effectue pas de transcodage (T1 T0 = 01). Quatre sérialisateurs spécialisés chacun en R, V, B, S

et commandés par Φ point = 4 MHz (Φ1 Φ0 = 11) vont délivrer en correspondance de R, de V, de B et de S les informations de teinte du point considéré selon l'effet de masquage ou de priorité suivant:

Sortie R = R

Sortie V = V $\overline{R}$

Sortie B = B $\overline{V}$ $\overline{R}$

Sortie S = S $\overline{B}$ $\overline{V}$ $\overline{R}$

Programme représentatif du mode:

Initialisation de la RAMA et de la RAMB
pour un écran uniforme (cf. bit-map 4 couleurs)

```
        LDA     #$3F        Passage du mode
        STA     $E7DC       "

DEB     LDB     #$0F        RAMA    RAMB
        STB     $5000       0 0     0 S

        JSR     ATT

        LDB     #$FF        RAMA    RAMB
        STB     $5000       0 0     B S

        JSR     ATT

        LDA     $E7C3       Commutation en RAMA
        ORA     #$01        "
        STA     $E7C3       "

        LDB     #$0F        RAMA    RAMB
        STB     $5000       0 V     B S

        JSR     ATT

        LDB     #$F0        RAMA    RAMB
        STB     $5000       R 0     B S

        JSR     ATT

        ANDA    #$FE        Commutation en RAMB
        STA     $E7C3       "

        JMP     DEB         Reprise
```

```
ATT     JSR     $E806       Boucle d'attente du
        CMPB    #$20        caractère espace
        BNE     ATT         "
        RTS                 "
```

Ce programme met en évidence le principe de superposition appliqué dans ce mode de fonctionnement. En configuration de base, après avoir tracé un GPL en noir sur fond gris, en appuyant sur la touche espace le même GPL devient bleu clair (S masqué); en appuyant encore sur la touche espace, le GPL devient vert pâle (S et B masqués) puis en appuyant de nouveau il devient rose (S, B, V masqués). Le fait de relancer à nouveau l'expérience en appuyant plusieurs fois sur la barre espace, démontre que S et B restent définitivement masqués par la présence de R ou V dans la RAMA.

Circuit de palette

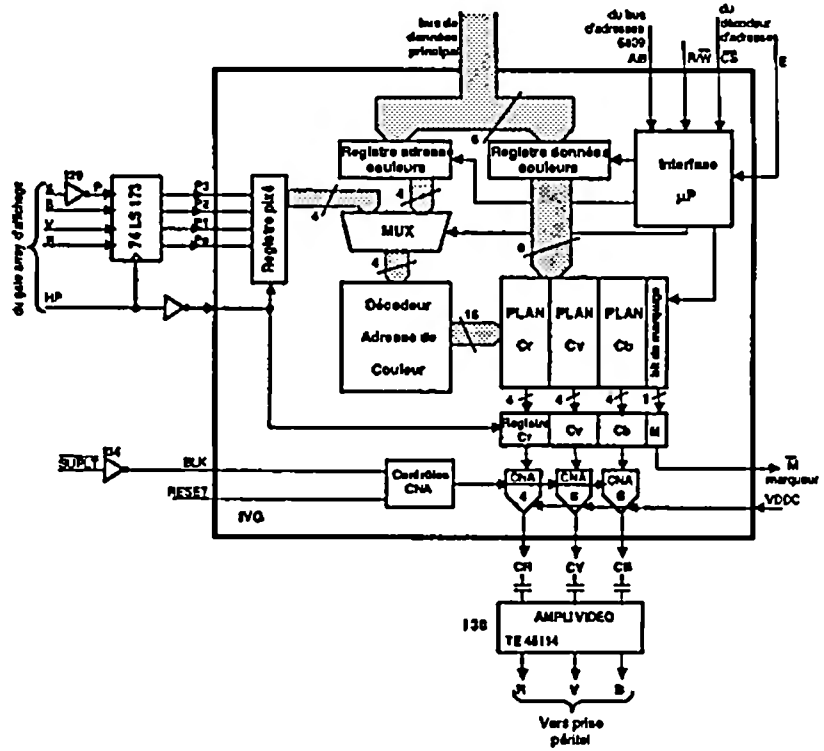


Figure 9. Circuit de palette. Utilisation de l'IVG 9369

Le système de palette I-28 utilise le circuit IGV EF 9369 de chez EFCIS. Son principal rôle est de pouvoir augmenter le nombre de teintes parmi les seize disponibles en configuration de base, sans pour autant intervenir sur les plans RAMA et RAMB. Il permet un choix de seize teintes parmi une palette de 4 096.

Description fonctionnelle de l'IGV

Il remplit les fonctions suivantes:

– Mémoire de couleur réinscriptible.
– Conversion numérique analogique des fondamentales R, V, B et mise aux normes péritélévision.
– Interfaçage avec le microprocesseur.

• *Mémoire de couleur*
C'est une mémoire vive de 16 mots de 13 bits (4 bits par fondamentale + 1 bit de marquage). Elle est accessible en adressage de deux manières:

– En lecture seulement par le système graphique (gate array d'affichage) à travers un latch 4 bits 74 LS 173 et un inverseur pour transformer le bit de saturation en bit pastel. La synchronisation ou validation est effectuée par l'horloge HP (image de Φ point). De cette manière, le circuit effectue la transposition des informations couleurs entre la mémoire écran et la prise péritel.

– Prioritairement, en lecture-écriture, par le 6809 E via l'interface microprocesseur, permettant ainsi de reconfigurer les couleurs et de fixer la couleur de transparence pour l'incrustation par l'intermédiaire du bit de marquage M (sortie marqueur).

• *Convertisseur numérique analogique*

Il délivre seize niveaux de tension différents par fondamentale (sorties CR, CV, CB) ce qui permet une gamme de 16 × 16 × 16 = 4 096 teintes. Les tensions de sorties s'échelonnent entre 0,8 V pour le niveau le plus bas et 1,8 V pour le niveau le plus haut. Ces tensions sont amplifiées de 6 dB par le circuit I-30 (TEA 5114) avec un OFFSET de 2 V pour attaquer la prise SCART.

Un contrôleur de CNA reçoit la commande de blanking (SUPLT) en provenance du gate array principal, fixant le niveau du noir pendant le retour ligne et trame. La commande de RESET force les sorties au niveau de noir jusqu'à l'accès microprocesseur suivant.

• *L'interfaçage* avec le 6809 E se fait par la commande CSN active. Ainsi, pour A0 = 0, le bus de données principal communique avec le registre de données couleurs représentant le contenu du plan mémoire. Pour l'entrée A0 = 1, le bus de données communique avec le registre d'adresses couleurs du plan mémoire. Le registre d'adresses couleurs offre alors la possibilité d'un adressage auto-incrémentable modulo 32 (écriture ou lecture du plan mémoire entier en 33 cycles micro).

Ainsi, un accès au registre de données incrémente automatiquement le registre d'adresse et, lors d'une programmation complète de la palette, il n'est pas nécessaire de venir réécrire le registre d'adresses.

Programmation de la palette

– La programmation du registre de données (désignation PALETTE) se fait à l'adresse E7DA en deux écritures ou deux lectures de 2 × 8 = 16 bits (dont 13 bits actifs), par auto-incrémentation du registre d'adresses selon la forme:

Première adresse	V V V V fondamentale V	R R R R fondamentale R
Deuxième adresse auto-incrémentée	X X X M bit de marquage	B B B B fondamentale B

et en sachant que:

RRRR = 1111 représente la fondamentale "rouge"
RRRR = 0000 représente le niveau du noir

VVVV = 1111 représente la fondamentale "vert"
VVVV = 0000 représente le niveau du noir

BBBB = 1111 repésente la fondamentale "bleu"
BBBB = 0000 représente le niveau du noir.

Toutes les autres combinaisons représentent des couleurs intermédiaires en intensité et saturation.

En relation avec le registre de données, la programmation du registre d'adresses couleurs (désignation PALETTE + 1) se fait à l'adresse CPU E7DB par une écriture ou une lecture du numéro de couleur qu'il faut veiller à transformer en adresse de plan mémoire sur 8 bits (une adresse couleur représentant deux adresses physiques successives par auto-incrémentation), selon la forme:

Adresse couleur ou numéro de couleur	Adresse plan mémoire
0	00
1	02
2	04
3	06
4	08
5	0A
6	0C
7	0E
8	10
9	12
A	14
B	16
C	18
D	1A
E	1C
F	1E

Exemple simple de programmation de la palette:

```
LDA     #$03        Désignation de la couleur jaune
ASLA                Transformation d'adresse (2 x 3)
STA     $E7DB       Stockage dans le registre d'adresses

LDD     #$0FFF      Choix du blanc saturé
STB     $E7DA       Stockage dans le registre de
STA     $E7DA       données avec auto-incrémentation

SWI
```

Ce programme permet de fixer la couleur numéro 3 en blanc, ce qui revient à dire que l'on transforme le jaune de la configuration de base en blanc.

Circuit d'incrustation

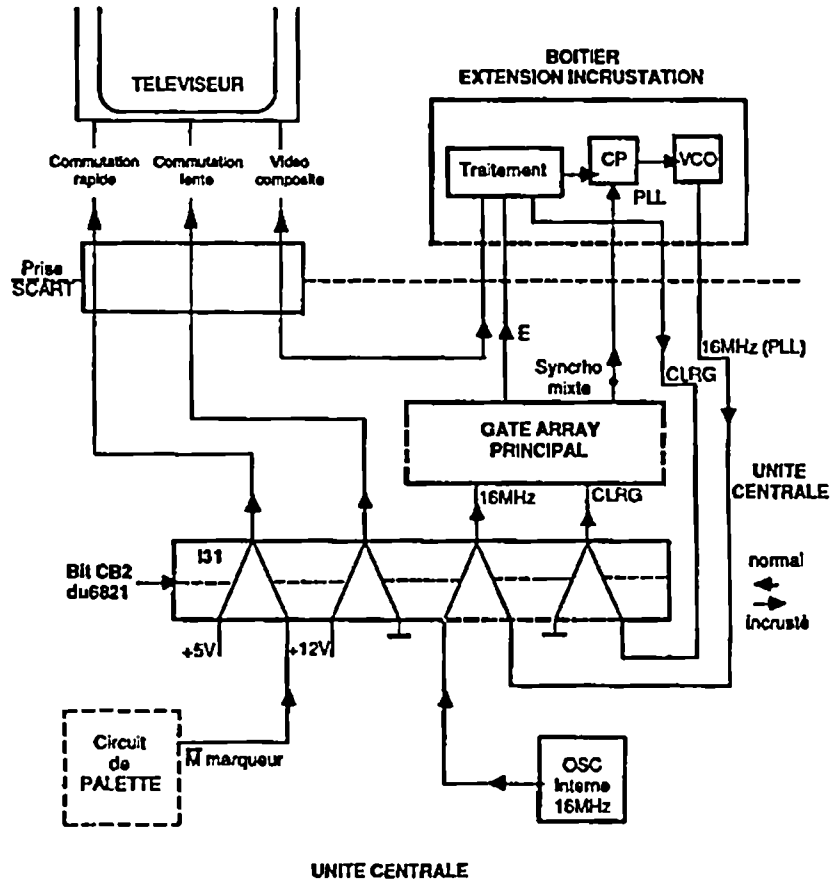


Figure 10. Circuits internes d'incrustation

Le but de l'incrustation est de superposer une image micro ordinateur sur une image vidéo (antenne ou magnétoscope). Le principe général du système est de permuter, par logiciel, la commande de commutation rapide (distribution des sorties RVB) de façon à ce que, pour toute génération d'images de la part du micro-ordinateur, pour chaque couleur dont le bit de marquage est positionné, l'image vidéo vienne se placer en substitution.

Il est aussi nécessaire d'asservir la synchronisation ligne du TO9 sur celle de la source vidéo (utilisation d'un VCO et d'une boucle à verrouillage de phase). De même, l'ordinateur fonctionnant en 624 lignes, il est indispensable de rattraper l'équivalent d'une demi-ligne pour chaque trame, afin de rester en concordance avec la source vidéo fonctionnant en 625 lignes (génération d'un signal de remise à l'heure CLRG).

Le VCO, le PLL et la commande CLRG sont situés dans un boîtier d'extension "incrustation". La figure 10 regroupe les principaux circuits internes à l'unité centrale, en relation avec l'incrustation.

Le passage en mode incrusté est réalisé par la commutation du bit CB2 (broche 12 du 6821, circuit I-43). Le signal obtenu, calé en phase avec E, vient commander un aiguilleur (multiplexeur 1-2 à 4 voies).

Ainsi, pour CB2 = 1:
Le système est en mode normal. Il reçoit les 16 MHz de l'oscillateur interne. Il génère du + 12 V pour la commutation lente, du + 5 V pour la commutation rapide.

Pour CB2 = 0:
Le système est en mode incrusté. Il reçoit les 16 MHz en provenance du VCO dans le boîtier d'incrustation. Il reçoit la commande CLRG activant l'entrée SYCL2 du gate array principal (remise à l'heure des compteurs lignes et trames). La commutation rapide est modulée par le marqueur (image vidéo pour l'état 0). La commutation lente est supprimée, la synchronisation mixte de l'ordinateur étant en phase avec la synchronisation TV. Le son du téléviseur est alors commuté.

6. Les interfaces parallèles

Le TO9 utilise deux circuits fondamentaux: le 6846 et le 6821 avec un environnement bien spécifique.

Utilisation du 6846: circuit I-41

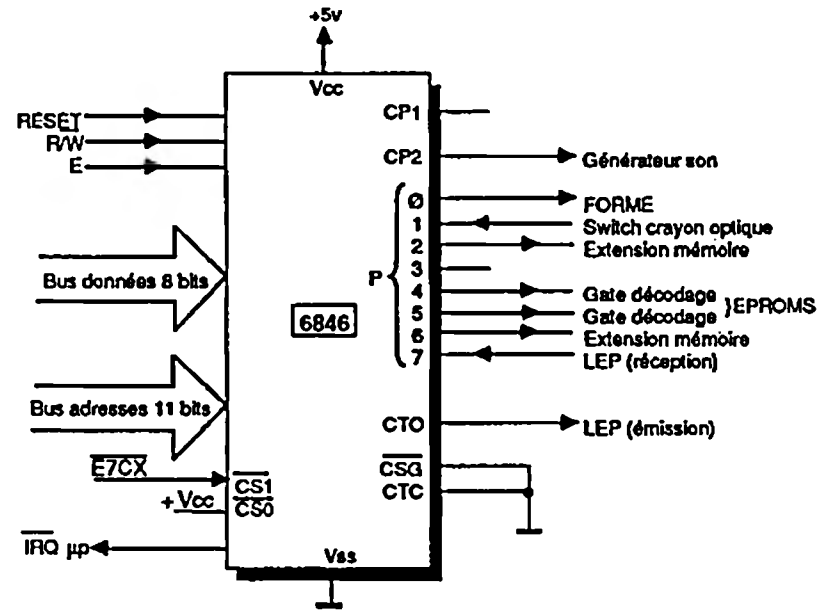


Figure 11. Le 6846 dans le TO9

Description fonctionnelle

– Le port 8 bits du 6846, appelé port C, a par initialisation cinq lignes P0, P2, P4, P5, P6, configurées en sorties et deux autres P1, P7, en entrées. Le bit P3 n'est pas utilisé.

P0 représente le bit de forme pour la commutation des plans mémoire RAMA ou RAMB.

P1 reçoit l'information de l'interrupteur du crayon optique. Un niveau 1 est présent lorsqu'il est activé.

P2 et P6 sont employés pour la commutation des banques de la mémoire extension disque virtuel.

P4 et P5 sont utilisés pour la commutation des EPROMS à travers le gate array de décodage d'adresses I-47.

P7 reçoit les informations numériques décodées, en lecture, du magnétophone LEP. Ce bit sert de test pour le programme d'initialisation de façon à savoir si le magnétophone est présent ou non (niveau 1 lorsqu'il est branché moteur arrêté).

– La ligne de contrôle CP2 est employée seule. Initialisée en sortie elle assure la génération du son, via le mélangeur.

– Le TIMER a un double rôle: en fonctionnement normal, il sert, par des demandes d'interruptions successives (sortie IRQ) toutes les 100 ms, à commander le clignotement du curseur. En utilisation du LEP, il code et fournit les informations numériques à enregistrer sur le magnétophone par la sortie CT0. Les informations digitales sont codées en salve de fréquences:

5 périodes à 4,5 KHz pour le bit 0
7 périodes à 6,3 KHz pour le bit 1

Ce procédé effectue une transmission sérialisée asynchrone à 900 bauds.

Adresses des registres internes

E7C0 - registre d'état composite (CSR)
E7C1 - registre de contrôle périphérique (CRC)
E7C2 - registre de direction de données (DDRC)
E7C3 - registre de données périphériques (PRC)
E7C4 - registre d'état composite (CSR)
E7C5 - registre contrôle temporisateur (TCR)
E7C6 - registre temporisateur d'octet de poids fort (TMSB)
E7C7 - registre temporisateur d'octet de poids faible (TLSB)

Utilisation du 6821 dans le TO9: circuit I-43

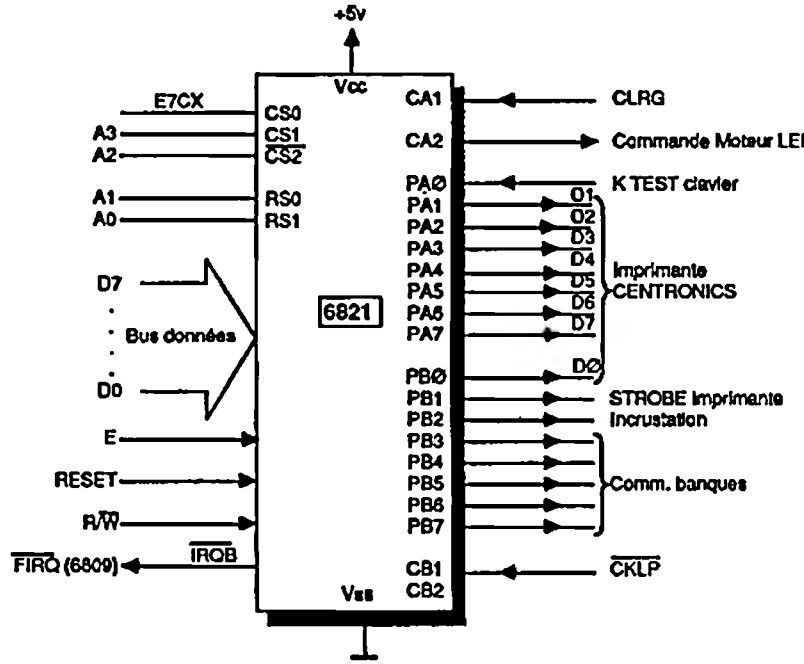


Figure 12. Le 6821 dans le TO9

Description fonctionnelle

Le 6821 sert à l'interfaçage d'une imprimante du type huit bits "parallèle" en mode CENTRONICS. Sept bits du port A sont utilisés à cet effet ainsi que deux bits du port B. Les lignes, configurées en sortie, servent au transfert des données sur huit bits et la commande de prise en compte STROBE (en PB1) qui est active à l'état 0. Les routines de gestion imprimante autorisent une émission de données, à condition que le signal "BUSY" en provenance de l'imprimante ne soit pas actif, c'est-à-dire, à l'état 0. Un test est effectué à partir de l'entrée CTSN de l'ACIA 6850 (circuit I-54).

La ligne PA0 reçoit une information KTEST en provenance du clavier. Ce dernier envoie un état 1 durable à chaque appui sur une touche. Ce bit est repris par la routine moniteur KTST permettant une vérification rapide d'une demande clavier (système anti-rebond).

PB2 est la commande d'incrustation active à 0. Elle reste à l'état bas lorsque l'utilisateur passe en mode incrustation.

PB3 à PB7 sont les bits de commutation de banque mémoire. Le programme d'initialisation fixe PB3 en sortie à l'état 0 et PB4, PB5, PB6, PB7 en entrée à l'état 1 par la présence de résistances de PULL UP (cf. Système de mémorisation).

Les lignes de contrôle ont chacune une application bien spécifique:

– CA1 reçoit le signal de remise à l'heure de l'éventuel boîtier d'incrustation. Il sert uniquement à positionner le *flag* du registre de contrôle CRA afin d'opérer une protection lorsqu'une demande d'incrustation est effectuée sans le boîtier.

– CA2 est en sortie. Active à l'état 0, elle permet de télécommander la mise en route du moteur LEP.

– CB1 reçoit la demande d'interruption lorsqu'une visée est effectuée sur l'écran par l'intermédiaire du crayon optique. La routine moniteur GETLTP prend en compte cette demande et démasque la sortie IRQN pour la transmettre en FIRQN sur le 6809 (cf. Gestion du crayon optique).

– CB2 n'est pas utilisé.

Adresses des registres internes:

E7C8 - registre de direction de données ou registre de données partie A (PRA)
E7C9 - registre de direction de données ou registre de données partie B (PRB)
E7CA - registre de contrôle partie A (CRA)
E7CB - registre de contrôle partie B (CRB).

7. La gestion du clavier et des périphériques

Le clavier est géré à partir d'un microprocesseur monochip du type 6805. Ce dernier est en liaison série avec l'unité centrale par l'intermédiaire d'un circuit d'interface série: l'ACIA 6850.

Utilisation du 6850 dans le TO9

La figure 13 (page suivante) met en évidence le rôle du 6850 en relation avec le microprocesseur du clavier.

Description

De par le décodage d'adresses envoyé sur l'entrée CS2 et la commande lecture-écriture R/WN, la situation des registres internes est telle que:

Adresse	R/WN	Registre sélecté
E7DE	0	CR (contrôle)
E7DE	1	SR (état)
E7DF	0	TDR (émission)
E7DF	1	RDR (réception)

L'horloge est commune pour l'émission et la réception (TXC et RXC reliés ensemble). Cette horloge oscille à une fréquence d'environ 145 KHz après division du 16 MHz par un compteur asynchrone dans un rapport $16 \times 7 = 112$.
Le registre de contrôle est programmé pour une prédivision interne de 16, ce qui permet une transmission à 9 600 bauds. L'unité centrale échange des informations avec le microprocesseur (monochip) du clavier dans les deux sens, soit en émission par TXD, soit en réception par RXD.

Avant la prise en compte d'un caractère envoyé, le logiciel de gestion clavier, fait un test rapide du bit PA0 du 6821 qui reçoit la commande KTEST. Ce bit passe à 1, lorsque l'on appuie sur une touche. La routine moniteur KTST force alors le bit C du registre d'état 6809 E à 1.

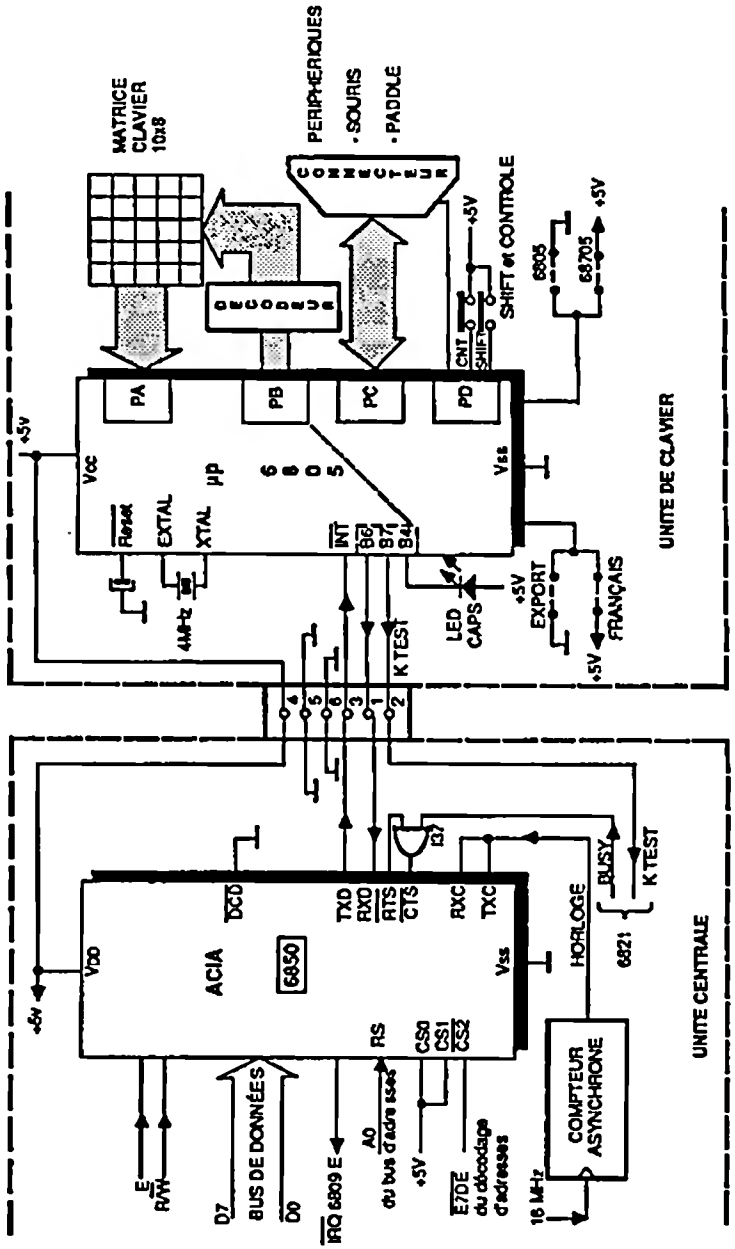


Figure 13. Structure matérielle du fonctionnement du clavier dans le TO9

En fonctionnement normal, le registre de commande est configuré pour maintenir RTSN à l'état 0, ce qui impose, via la porte I-37, CTSN = 0 et indique par là-même que le périphérique est toujours prêt à émettre.

En attente d'appui sur une touche, le contenu du registre d'état est de 02, précisant notamment que les registres de réception et d'émission sont vides.

Le 6850 a un rôle annexe, consistant à récupérer le signal BUSY de l'imprimante CENTRONICS commandée par le 6821. En gestion imprimante parallèle, les routines correspondantes forcent RTSN au niveau 1 et viennent alors tester l'état de CTSN dans le registre SR.

Le clavier

Présentation

L'ensemble clavier du TO9 est séparé de l'unité centrale et y est relié par un cordon spirale à 6 conducteurs. Ce clavier comporte 81 touches à répétition automatique et est muni d'une prise SUB-9 broches permettant de connecter les périphériques d'entrée, tels que souris et potentiomètres de commande *(paddle)*.

Il est équipé d'un microprocesseur monochip du type 6805, chargé de la scrutation des touches, de la reconnaissance et du traitement de données fournies par les périphériques. Le monochip est commandé par une horloge autonome à 4 MHz.

Les touches du clavier sont organisées en matrice de 10 lignes et 8 colonnes. Les 10 lignes sont commandées, via un décodeur (4 vers 10), par le port B du microprocesseur. Les 8 colonnes sont appliquées sur le port A du microprocesseur.

Lorsqu'une touche est pressée, une liaison est établie entre une ligne et une colonne. Le microprocesseur détecte ce changement d'état et positionne alors les 10 lignes à 0 V, l'une après l'autre. Lorsque la ligne concernée passe à 0, la colonne passe également à 0 et le microprocesseur connaît alors l'emplacement de la touche.

Les touches shift (⇧) et contrôle (CNT) sont appliquées directement sur le port d'entrée D.

Le voyant associé à la touche verrouillage majuscule (CAPS) est commandé par une sortie de port B qui est capable d'alimenter directement une diode LED (I = 10mA).

Les périphériques sont reliés par l'intermédiaire de la prise correspondante au port C.

Un strap permet de monter, soit un microprocesseur 6805 masqué, soit un microprocesseur 68705 (utilisation d'une EPROM). Un autre strap permet de choisir la configuration du clavier, français ou export.

Signaux échangés avec l'unité centrale

• Signaux reçus par le clavier

L'unité centrale est capable de modifier certaines fonctions du clavier, telles que réinitialisation, commande majuscule-minuscule, autorisation de périphériques, en envoyant un mot série (UC -> Clavier).

Ce mot est constitué par:

– 1 bit de start
– 3 bits de données
– 1 bit de stop.

La vitesse de transmission est de 9 600 bauds.

L'unité centrale utilisant l'ACIA pour envoyer ces commandes, le code est donc constitué de 8 bits. Seuls les 3 bits de poids faible sont utilisés, les autres sont toujours à l'état haut.

• Signaux émis vers l'unité centrale

Le clavier envoie vers l'unité centrale, des informations sous forme d'octets, transmis en série avec le format suivant:

– 1 bit de start
– 8 bits de données ou octet (code ASCII)
– 1 bit de parité (indicateur)
– 3 bits de stop.

La vitesse de transmission est de 9 600 bauds.

Le bit de parité sert d'indicateur pour différencier les codes clavier des codes périphériques. Les 3 bits de stop permettent d'obtenir un écart de temps minimum entre deux mots consécutifs. Les informations sont transmises de la manière suivante:

• Mode clavier seul

Un octet est envoyé à chaque appui d'une touche. Si cette touche reste enfoncée, après un délai de 0,8 secondes, le code est envoyé toutes les 70 milli-secondes (répétition automatique). Si deux touches sont pressées simultanément, les deux codes sont envoyés successivement. Dans ce cas, la répétition automatique est inhibée, et ceci tant que les deux touches (ou plus de deux) sont enfoncées.

• Mode périphérique

Lorsqu'un périphérique est connecté et actif, les informations transmises à l'unité centrale sont composées de trains de quatre octets successifs qui se répètent toutes les 10 milli-secondes. Ce train est organisé de la façon suivante:

– 1er octet: Code clavier

Cet octet vaut 00, si aucun code clavier n'est envoyé. La parité de cet octet est impaire (bit de parité = 1, si le nombre de bits à 1 est impair). Si le clavier est en répétition automatique, le code est envoyé toutes les 70 milli-secondes, c'est-à-dire, une fois tous les 7 trains.

– 2ème octet: Valeur du déplacement (boule ou potentiomètre) en X

Cet octet varie de 00 à $FF. La parité est paire.

– 3ème octet: Valeur du déplacement (boule ou potentiomètre) en Y

Cet octet varie de 00 à $FF. La parité est paire.

– 4ème octet: Dépassement et gachettes

La parité est paire. La désignation est la suivante:

bit 7: dépassement Y
bit 6: dépassement X
bit 5:
bit 4:
bit 3:
bit 2: gachette 2
bit 1:
bit 0: gachette 1

En conclusion, le bit de parité utilisé comme indicateur est impair uniquement pour le code clavier. Cela permet la synchronisation des données à la réception par l'unité centrale.

8. Gestion du crayon optique

Fonctionnement du crayon optique

Le crayon optique est constitué de deux éléments séparés.

– Un interrupteur de validation tactile.
– Un phototransistor de détection optique du spot sur l'écran.

Son action est dirigée à partir de diverses routines situées dans le moniteur (voir page 200).

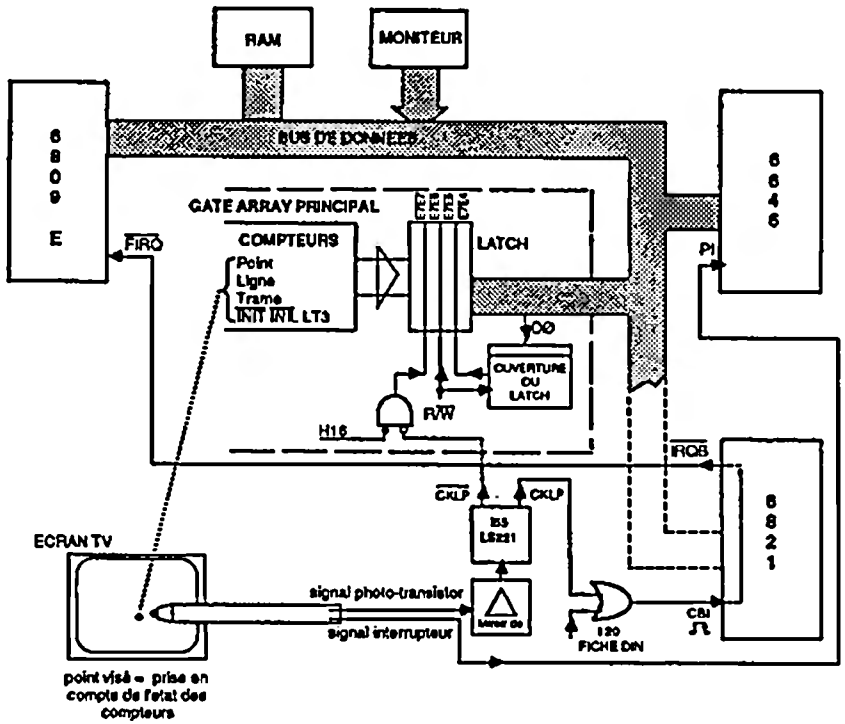


Figure 14. Synoptique du fonctionnement du crayon optique dans le TO9

Fonctionnement de l'interrupteur

En effectuant une pression sur l'écran avec le crayon optique, on ferme l'interrupteur de ce dernier sur la tension d'alimentation + 5 V. Cet état haut résultant vient informer le bit P1 du port du 6846.

En logiciel, ce bit à 1 est repris par la routine LPINT du moniteur qui, après un test "anti-rebond", force le bit de carry du 6809 conjointement à 1.

Fonctionnement du phototransistor

Lorsque l'écran du téléviseur reçoit, via la prise péritel, une information RVB, il n'est conçu, au même instant, qu'un point sur l'écran. S'il est dans la fenêtre de visualisation, ce point appartient à un GPL dont l'adresse est présente, toujours à cet instant, dans les compteurs lignes et trame du gate array principal. Si face à ce point, on place le phototransistor, ce dernier va émettre une information récupérée en tension, amplifiée (circuit à miroir de courant) et mise en forme par un monostable (circuit I-55), d'où résulte une impulsion calibrée à 700 ns. Cette impulsion est aiguillée en deux directions CKLPN et CKLP.

• *Action de CKLPN = $\overline{CKLP}$*

CKLPN attaque le gate array principal dans le but de latcher l'état des compteurs points, lignes et trame. Pour ce faire, le signal est découpé par la fréquence de l'horloge mère à 16 MHz, ce qui permet la reconnaissance des compteurs points. En fait, l'opération générale du stockage des compteurs points, lignes et trame, ne pourra se faire que par un ordre du 6809. Le CPU doit envoyer en écriture, par l'intermédiaire du bit D0 à l'état 1 et à l'adresse E7E4, une demande d'ouverture ou de prise en compte dont la finalité est de rendre CKLPN actif. Cette opération doit être effectuée avant l'arrivée de ce signal.

• *Action du CKLP*

CKLP, via une porte "OU" susceptible de recevoir des informations d'un éventuel codage à barres, correspond à une demande d'interruption sur l'entrée de la ligne de contrôle CB1 du 6821. Après autorisation, cette demande d'interruption est dirigée sur la borne FIRQN du 6809. La sous-routine d'interruption correspondante "INTERLP" du moniteur assure la prise en compte, en lecture du latch gate array principal, de l'état des compteurs. La lecture s'effectue sur les quatre adresses des flip-flop E7E4 E7E5 E7E6 E7E7 selon la forme suivante:

Adresses du latch	Bus de données en correspondance D7	D6	D5	D4	D3	D2	D1	D0
E7E4	T12	T11	T10	T9	T8	T7	T6	T5
E7E5	T4	T3	TL2	TL1	TL0	E	H2	H4
E7E6	LT3	INILN	0	0	0	0	0	0
E7E7	INITN	INITN	0	0	0	0	0	1

indépendant de CKLP (non latché)

• *Action générale*

Aux adresses E7E4 - E7E5, la connaissance de l'état de l'ensemble des compteurs points, lignes et trames (H4 à T12) permet à la routine principale GETLTP (routine appelée par l'utilisateur) de transcrire, pour le point visé, l'acquisition effectuée en coordonnées X et Y de la fenêtre de visualisation. Cette routine applique la formule suivante:

Y = acquisition/320
X = reste de la division

A l'adresse E7E6, INILN indique si la mesure du crayon optique s'est effectuée dans la partie horizontale active de la fenêtre ou dans les bords droit ou gauche. A la même adresse, LT3 permet, dans le cas d'une mesure horizontalement en dehors de la fenêtre, de distinguer le bord droit ou gauche du tour de l'écran.

A l'adresse E7E7, INITN, latché par CKLP (bit de poids 6), est testé dans la sous routine INTLP pour savoir si la visée s'est effectuée dans la portion verticale active de la fenêtre de visualisation ou dans les bords supérieur ou inférieur du tour de l'écran. A la même adresse, INITN, (bit de poids 7) non latché par CKLP et image du signal de construction de la fenêtre en trame, permet à la routine principale GETLTP d'attendre la remise à zéro des compteurs pour tenter une acquisition.

Selon la distance du crayon optique à l'écran, le système prévoit une prise en compte d'un nombre plus ou moins grand de mesures (8 au maximum, 2 au minimum), les résultats étant placés dans un buffer.

9. L'exploitation du lecteur-enregistreur de disques

Description

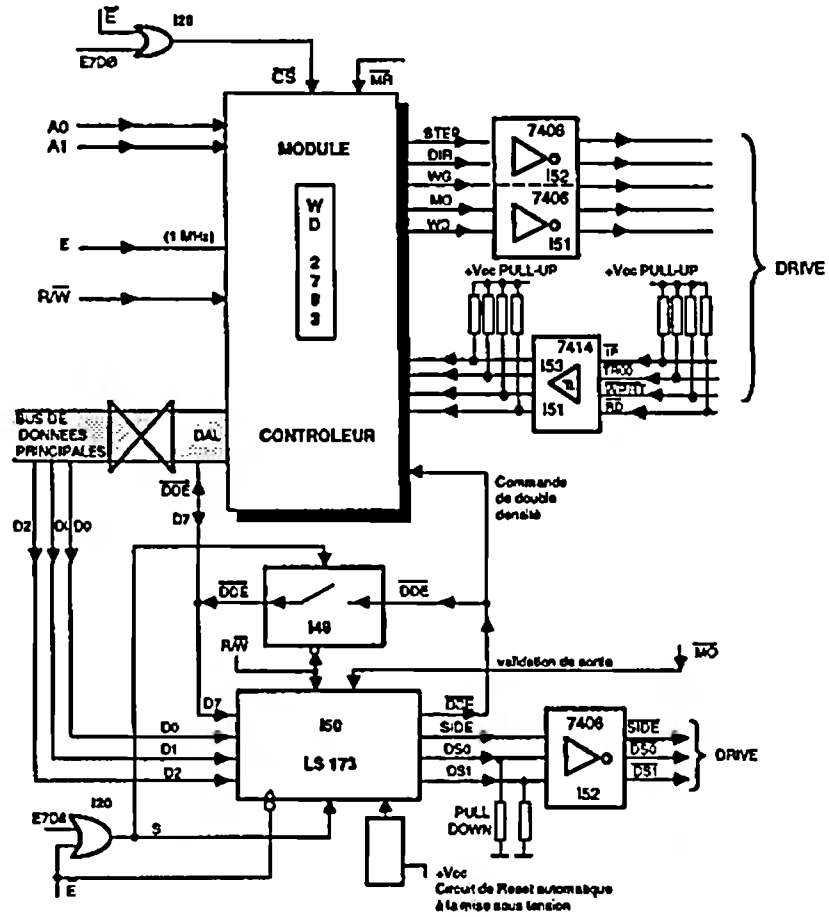


Figure 15. Contrôle du lecteur-enregistreur de disquettes dans le TO9

L'appareil est composé, sous forme d'un module approprié, par un circuit d'interface, contrôleur de disques WD 2793 (WESTERN DIGITAL). Il est destiné à adapter les caractéristiques électriques et mécaniques du lecteur-enregistreur de disquettes, (DRIVE ou FLOPPY) 3,5 pouces, à l'unité centrale. Il remplace avantageusement le WD 1770 dont les premières versions du TO9 étaient pourvues.

La figure 15 (page précédente) représente et schématise les différentes commandes liées au CPU et au DRIVE. Ces commandes peuvent être répertoriées selon la description suivante:

– Liaison CPU

$\overline{CS}$	assure la sélection du boîtier et, à travers l'action d'un monostable interne au module, la mise en marche du moteur du lecteur (commande MO).
A0 et A1	permettent la sélection des registres internes.
R/WN	est la commande de lecture-écriture.
DAL	constitue le bus de données bidirectionnel, en relation avec le CPU, pour faire transiter les mots de commande et d'état.
E	assure le synchronisme du transfert des données.

– Commandes mécaniques

En sortie:

$\overline{STEP}$	commande du moteur pas à pas.
$\overline{DIR}$	commande de direction du moteur pas à pas.
$\overline{WG}$	commande de validation d'écriture.
$\overline{MO}$	commande du moteur d'entraînement de la disquette.

En entrée:

$\overline{IP}$	détection d'index.
$\overline{TROO}$	détection de la piste 0.
$\overline{WPRT}$	détection de la protection en écriture.

– Transfert des données

$\overline{RD}$	ligne de transmission en lecture.
WD	ligne de transmission en écriture.

– Commande d'initialisation

$\overline{MR}$	MASTER RESET, en liaison avec le RESET du CPU.

Fonctionnement général

Le contrôleur est commandé par l'action de la porte OU (I-20) et de celle du décodage d'adresses combiné avec A0 et A1. Ainsi, il est sélecté pour E = 1 aux quatre adresses E7D0, E7D1, E7D2, E7D3.

Un monostable redéclenchable (74 LS 122) interne au module, activé par le décodage d'adresses, assure la mise en marche du moteur pendant tout le temps ou le 6809 travaille en relation avec le lecteur. La commande résultante de mise en marche du moteur vient informer l'entrée "READY" du contrôleur.

– Le latch I-50 (74 LS 173) associé à un ensemble de portes tristates (I-49) réalise un registre de commande de drive et de choix de densité. De par le montage, ce registre est accessible en écriture à l'adresse E7D8 pour E = 1. L'opération d'écriture est réalisable sour la forme:

$\overline{DDE}$ X X X X DS1 DS0 SIDE

avec $\overline{DDE}$: Choix de densité

$\overline{DDE}$ = 1 simple densité

$\overline{DDE}$ = 0 double densité

avec DS1 et DS0 – commande de drive et SIDE commande de face.

La gestion du système présente la particularité suivante: Une face de disquette correspond à un numéro de lecteur. Ainsi, les numéros 0 et 1 font partie du drive intégré au TO9 (1 est inexploitable) 2 et 3 font partie d'un drive externe.

Tableau des commandes:

DS0 = 1	DS1 = 0	SIDE = 0	-> lecteur n°0
DS0 = 1	DS1 = 0	SIDE = 1	-> lecteur n°1
DS0 = 0	DS1 = 1	SIDE = 0	-> lecteur n°2
DS0 = 1	DS1 = 1	SIDE = 1	-> lecteur n°3

Le rôle de I-49 est de rendre le registre accessible en lecture pour l'information du choix de densité DDEN.

Les circuits I-51 et I-52, de structure à collecteur ouvert, combinés avec des résistances de pull up et de pull down, adaptent et bufferisent les différentes commandes entrée et sortie du WD 2793.

Troisième partie

Analyse matérielle du TO8

1. Analyse générale

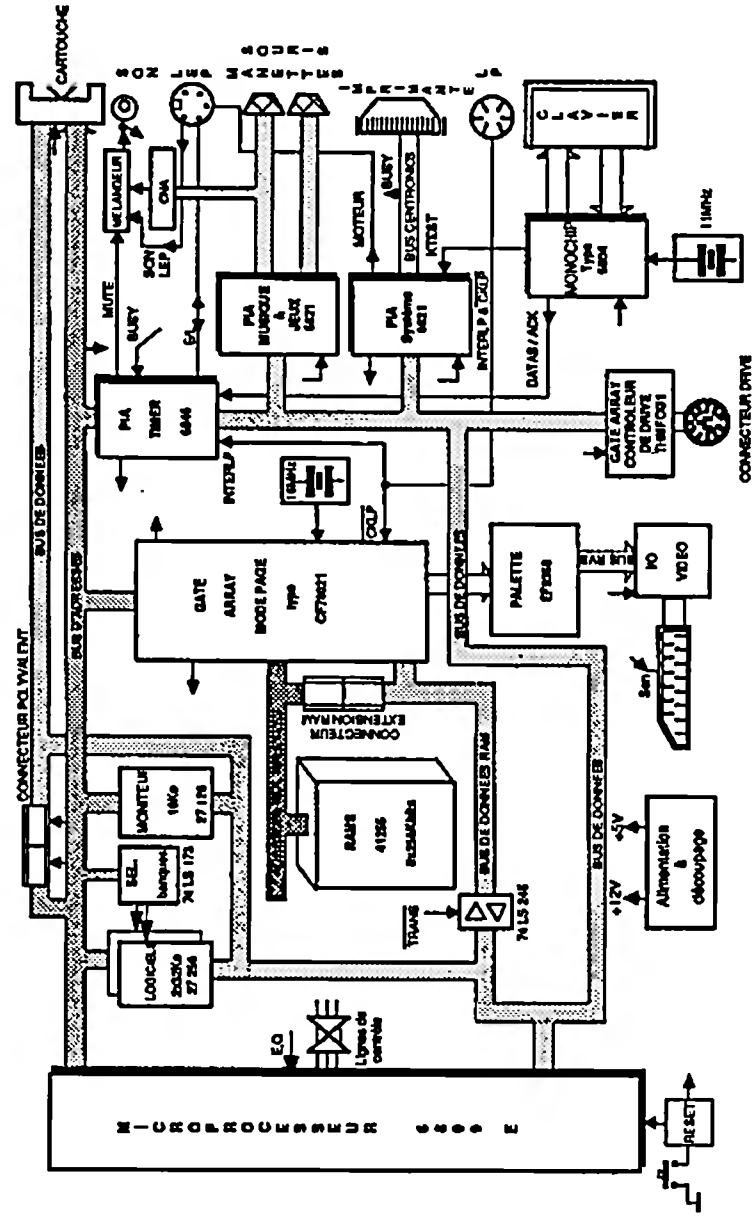


Figure 16. Synoptique de l'unité centrale TO8

Conception générale - Description

Le micro-ordinateur TO8 est conçu comme ses prédécesseurs, à partir d'un 6809 E, microprocesseur commandé à la fréquence de 1 MHz par deux horloges extérieures en quadrature: *E* et *Q*.

Le bus d'adresses 16 bits est direct et permet d'accéder aux différentes mémoires mortes et aux registres.

Le bus de données 8 bits véhiculant les principaux échanges a une ramification particulière: le bus de données RAM qui est bufferisé et relié par un 74 LS 245. Pendant la phase non active du 6809, ce bus est isolé du bus de données principal par l'action du 74 LS245 commandé par le signal TRANS. C'est durant cette période que le gate array accède aux mémoires en "mode page".

Les lignes de contrôles du 6809 E correspondent:

– aux commandes de lecture écriture R/WN des différents registres et mémoires,

– aux demandes d'interruption IRQN concernant la gestion du clavier, le clignotement du curseur, les manettes de jeux et la souris,

– aux demandes d'interruption FIRQN pour le fonctionnement du crayon optique et du code barre.

Un circuit de réinitialisation "RESET" est en relation avec le 6809 E et différents circuits, tels que le contrôleur de clavier 6804.

• *Pour la mémoire morte*

Les deux pages de 6 Ko du moniteur ainsi que les deux pages de 1,9 Ko de logiciel contrôleur de disque sont logées dans une EPROM 16 Ko 27 128.

Les 2 × 32 Ko de logiciel d'application BASIC 1 - divers - BASIC 512 - EXTRAMONITEUR, sont logés chacun dans une ROM ou EPROM 32 Ko selon une répartition en quatre banques.

La commutation des banques s'effectue en **programmation** par une écriture d'adresses ROM dans un *latch* 74 LS 173. Ce circuit permet de sélecter chaque partie concernée dans une des mémoires mortes.

La cartouche de logiciel d'application externe est reliée à l'unité centrale par l'intermédiaire d'un connecteur. Elle est sélectée par logiciel à partir d'un bit de PIA du 6846.

• *Pour la mémoire vive*

Huit boîtiers 256 Kbits 41256 forment un plan mémoire de 256 Ko comprenant un découpage de pages logiques de 16 Ko.

Les différentes pages sont commutées par un adressage physique sur 18 bits en provenance du gate array mode page.

L'extension mémoire, accessible par un connecteur de carte relié au bus de données RAM, est constituée de huit boîtiers 41256 qui regroupent seize autres pages de RAM utilisateur (pages 16 à 31) adressées de la même manière et différenciées par le signal CAS2N.

Outre le 6809 E, l'élément vital de l'unité centrale est un circuit à réseau logique (gate array) appelé "mode page" car il permet, par un bus d'adresses multiplexées, de faire fonctionner les RAMS dynamiques 41256 dans le type de mode susnommé.

Pendant la phase non active du CPU, il assure le rafraîchissement des mémoires et de l'écran. Piloté par une horloge mère de 16 MHz, il délivre les différents signaux de timing et de commande vidéo. Il fabrique tous les décodages d'adresses. Il gère, en partie, le fonctionnement du crayon optique par le signal CKLPN. Il regroupe et produit les huit modes d'affichage et définit les couleurs du cadre. Il permet de définir la carte mémoire en quatre espaces logiques principaux:

L'espace "cartouche" de 0000 à 3FFF
L'espace "écran" de 4000 à 5FFF
L'espace "système" de 6000 à 9FFF
L'espace "données" de A000 à DFFF

Il autorise l'utilisateur à affecter une page de RAM à un espace logique selon certaines modalités; notamment, il permet de recouvrir 16 Ko de logiciel par une page de mémoire vive.

Un circuit de palette du type EF 9369, programmable par le 6809 E, permet sous la dépendance du mode d'affichage déterminé une variété de seize teintes exploitables directement par l'écran parmi un choix de 4 096 au total.

Il délivre les trois informations B, V, R reprises et adaptées par des circuits d'interfaçage vidéo comprenant, entre autres, le dispositif d'incrustation. Ces circuits, recevant des signaux de synchronisation et de blanking du gate array, fournissent les tensions et adaptations nécessaires pour la prise péritel aux normes SCART ainsi que pour un éventuel codeur modulateur PAL dans la version export.

Un deuxième connecteur de carte, appelé polyvalent, permet de relier les bus et signaux nécessaires aux extensions.

Trois circuits d'interface sont en relation avec différents périphériques:

– Un 6846 qui assure par le timer l'envoi codé des informations numériques à enregistrer sur le magnétophone LEP (sauvegarde). Inversement, une ligne du PIA récupère les informations numériques décodées en provenance du LEP (chargement). Les autres lignes du PIA sont affectées à la commande de silence ou "MUTE" en utilisation de la souris, à la prise en compte du signal BUSY de l'imprimante, à la communication avec le 6804 pour la gestion du clavier, au traitement de l'information "tactile" du crayon optique LP (signal INTERLP) et, enfin, à la commutation des logiciels internes ou de la cartouche externe.

– Un 6821 "musique et jeux" qui est chargé par ses deux ports de huit bits et ses quatre lignes de contrôle de la génération du son et de la liaison manettes des jeux et souris. Un convertisseur numérique analogique CNA récupère le son synthétisé et l'envoie vers un circuit mélangeur recevant, par ailleurs, le son du LEP et la commande de MUTE. La sortie du mélangeur alimente la prise peritel et une prise auxiliaire CINCH.

– Un 6821 "système" qui assure principalement, par l'intermédiaire d'un connecteur spécialisé, la gestion d'une imprimante en mode parallèle de type CENTRONICS, prend en compte, par le signal KTEST, l'action d'une touche du clavier, procure les demandes d'incrustation et la télécommande du moteur LEP.

Un microprocesseur monochip du type 6804 sert de contrôleur de clavier. Piloté par une horloge oscillant à 11 MHz, il dialogue en transmission série codée avec le 6809 E par le truchement du 6846. La liaison est effectuée dans les deux sens à l'aide de deux fils DATAS et ACK. Le clavier est relié au 6804 par deux connecteurs assurant une transmission parallèle.

Un contrôleur de disquettes, sous forme d'un deuxième circuit à réseau logique ou gate array (THMFC01), programmable par le 6809 E, délivre par l'intermédiaire d'une prises DIN 14 broches les signaux nécessaires au fonctionnement d'un lecteur de disquettes.

Une alimentation à découpage au secondaire fabrique les deux tensions de + 5 V et + 12 V nécessaires à la configuration de l'unité centrale.

2. Le 6809 E dans le TO8

Suivant la lignée de ses prédecesseurs, le fonctionnement général du TO8 est basé sur le principe fondamental de la phase active et non active du 6809 E (cf. étude du TO9, paragraphe sur le Principe fondamental page 34).

La fréquence des horloges E et Q en provenance du gate array reste de 1 MHz et la constante de temps du RESET de 1 seconde.

La structure de bus est simplifiée. Un seul buffer 74 LS 245 est utilisé pour l'aiguillage du bus de données RAM (cf. synoptique). En dehors de l'action de R/WN déterminant le sens du tranfert des informations, le 74 LS 245 est commandé par le signal TRANSN réagissant en fonction de E, des zones adressées et indirectement du CSCRTN (cf. commutation des logiciels) selon la forme:

E	CSCRTN	zones adressées	TRANSN	
0	X	XXXX	1	
1	0	0000-3FFF	1	
1	1	0000-3FFF	0	} validation
1	1	4000-DFFF	0	} du buffer

La consultation de ce tableau amène les remarques suivantes:

Pour E = 0, le CPU n'a pas accès au bus RAM. Ce dernier est réservé au rafraîchissement.

Pour E = 1, selon la programmation du gate array mode page, la zone d'adresse 0000-3FFF initialement réservée pour les logiciels internes ou externes peut être attribuée à une page de RAM. Dans ce cas, CSCRN = 1 et le signal TRANSN = 0 valident le buffer permettant au microprocesseur d'accéder à la mémoire vive.

Exceptées les zones d'adresses du gate array mode page, l'espace E000-FFFF reste en majorité innaccessible par le CPU à travers le bus RAM.

Les deux entrées d'interruptions utilisées sont IRQN et FIRQN. La première sert à gérer le clignotement du curseur (sortie du TIMER 6846), le fonctionnement du clavier (PIA du 6846) ou d'un périphérique externe tel que manettes de jeux, souris (6821 musique et jeux). La deuxième gère l'action du phototransistor dans le crayon optique lors d'une visée (gate array mode page) ou, en relation avec un logiciel adapté, l'action d'un éventuel capteur de "code barre".

3. Gestion de la mémoire morte

Contrairement au TO7, au TO7/70 et d'une façon similaire au TO9, le TO8 est livré avec des logiciels intégrés qui sont adressés dans le même espace mémoire que la cartouche. Le moniteur, quant à lui, réside dans une autre partie de zone mémoire.

Description des logiciels

La mémoire morte interne du TO8 est répartie en trois boîtiers:

– Une ROM ou EPROM de 32 Ko (27256) organisée en deux pages de 16 Ko (banque 0, banque 1) contenant le BASIC 512 et l'EXTRAMONITEUR.

– Une ROM ou EPROM de 32 Ko (27256) organisée en deux pages de 16 Ko (banque 2, banque 3) contenant le BASIC 1, la page d'en-tête, le réglage palette et le DOS ICONIQUE.

– Une ROM ou EPROM de 16 Ko (27128) organisée en deux pages de 8 Ko contenant le MONITEUR de l'unité centrale et du lecteur de disquettes externe.

La cartouche enfichable de 16 Ko ou 32 Ko représente la mémoire morte externe.

Commutation des logiciels

La figure 17 décrit le mécanisme général de fonctionnement.

De par les décodages d'adresses CSCRTN et CSMN en provenance du gate array mode page, le système assure la sélection de la zone cartouche 0000-3FFF et zone moniteur E000-FFFF moins quelques adresses prises par les registres des circuits périphériques.

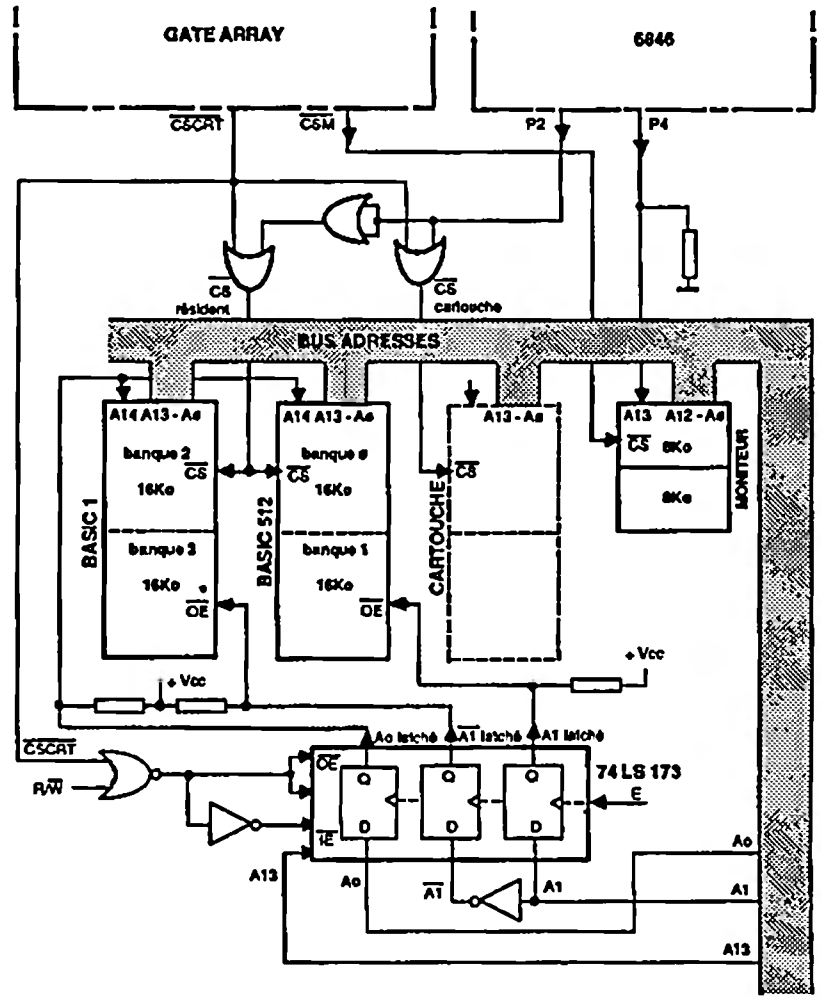


Figure 17. Gestion des ROMS dans le TO8

Sélection d'une page moniteur

Par construction, le bit P4 du 6846 étant relié au bit A13 du bus d'adresse de la mémoire de 16 Ko, ce bit permet de choisir la page de 8 Ko haute ou basse du boîtier. Ainsi:

P4 = 0 ==> partie basse de la ROM
P4 = 1 ==> partie haute de la ROM

La résistance reliée à la masse a pour effet, lors de l'initialisation, de rendre accessible automatiquement la partie basse de la ROM, P4 étant positionnée en entrée.

Sélection entre logiciels résidents et cartouche

La commutation se fait à partir du bit P2 du 6846 qui, combiné avec le décodage d'adresse CSCRTN, restitue à travers une logique câblée les deux commandes telles que:

$\overline{CS}$ résident = $\overline{\overline{CSCRT} + \overline{P2}}$

$\overline{CS}$ cartouche = $\overline{\overline{CSCRT} + P2}$

Ainsi, pour P2 = 0:
$\overline{CS}$ cartouche est actif dans le champ d'adresses 0000-3FFF et valide la cartouche.
$\overline{CS}$ résident est inactif.

Ainsi, pour P2 = 1:
$\overline{CS}$ résident est actif dans le champ d'adresses 0000-3FFF et valide les deux boîtiers de logiciels internes.
$\overline{CS}$ cartouche est inactif.

De par l'action du gate array mode page, ce montage offre la possibilité d'inhiber les logiciels dans l'espace mémoire 0000-3FFF qui leur est normalement alloué. Dans ce cas de figure, CSCRTN = 1 constant. Ce type de fonctionnement permet à l'utilisateur de substituer de la ROM par une page de RAM pouvant être, par ailleurs, chargée par un logiciel en provenance d'une mémoire de masse (cf. fonctionnement du gate array mode page, page 105).

Sélection des quatre banques de logiciels internes

Le mécanisme employé est comparable à celui utilisé par les cartouches (COLORCALC, COLORPAINT, BASIC II ...) et par le TO9 (en remplaçant le signal CSN par CSCRTN).

Synthèse de fonctionnement

Le tableau suivant résume le mécanisme général des commutations selon différents cas de figure:

Champ d'adr.	$\overline{CSCRT}$	$\overline{CSM}$	P2	P4	A1 latché	A0 latché	Logiciel sélecté
0000-3FFF	0	1	0	X	X	X	Cartouche
- -	0	1	1	X	0	0	Banque 0
- -	0	1	1	X	0	1	Banque 1
- -	0	1	1	X	1	0	Banque 2
- -	0	1	1	X	1	1	Banque 3
4000-DFFF	1	1	X	X	X	X	Néant
E000-E7AF E800-FFFF	1	0	X	0	X	X	Moniteur partie basse
E000-E7AF E800-FFFF	1	0	X	1	X	X	Moniteur partie haute

4. Les mémoires vives

Technologiquement, les 256 Ko de mémoire vive résidente du TO8 sont constitués par 8 boîtiers intégrés du type 41256. Ces circuits ont une capacité de 256K × 1 bit. Pour une reconstitution en données de 8 bits, huit 41256 sont associées, chacune étant spécialisée par un poids de D7 à D0 (cf. figure 18)

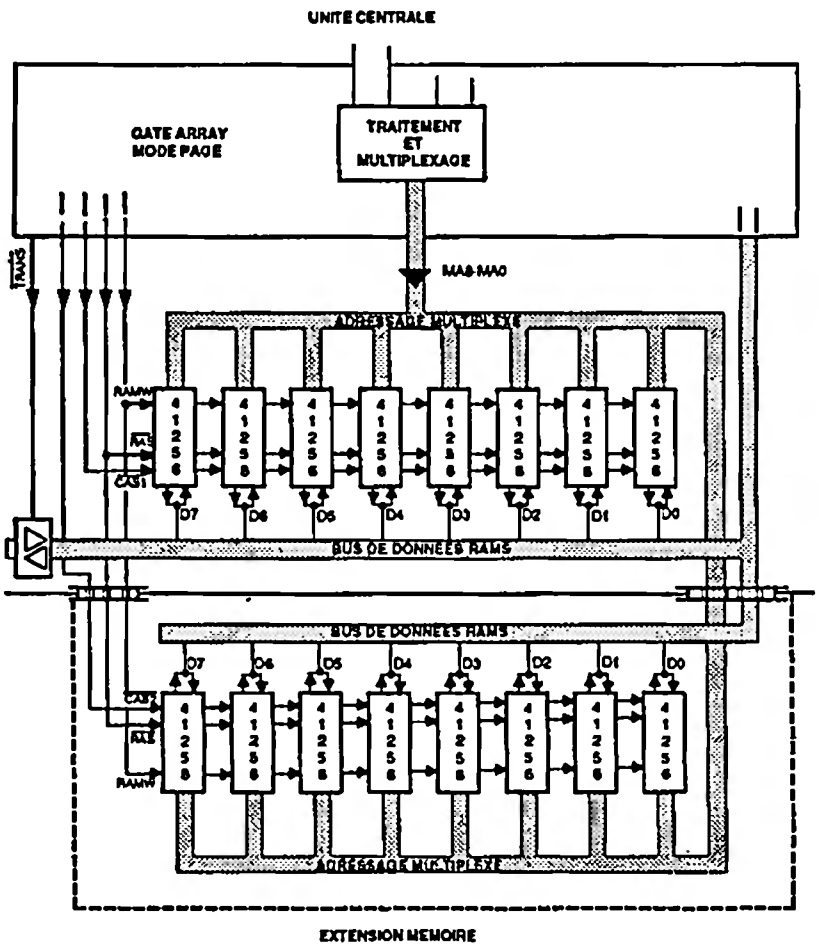


Figure 18. Système de mémorisation RAMS dans le TO8

Fonctionnement d'une 41256

Les 41256 sont des mémoires vives dynamiques qui permettent de stocker sous forme de matrice de $2^9 = 512$ lignes et de $2^9 = 512$ colonnes, $512 \times 512 = 256$ Kbits. L'adressage d'une telle matrice, à structure symétrique, nécessite 2×9 bits d'adressage envoyés en deux temps sur les 9 fils du bus d'adresse correspondant. Aux 9 bits de poids faible correspond l'adressage ligne validé par le signal de ligne RASN; aux 9 bits de poids fort correspond l'adressage colonne validé par le signal de colonne CASN. Le diagramme suivant précise la forme multiplexée de l'adressage:

a0	a9
a1	a10
a2	a11
a3	a12
a4	a13
a5	a14
a6	a15
a7	a16
a8	a17
Adressage ligne validé par RASN et rafraîchissement d'une ligne	Adressage colonne validé par CASN

On peut faire suivre un adressage ligne bien particulier par deux adressages colonnes successifs en utilisant deux validations de CASN, pour RASN actif constamment. Ce type de fonctionnement s'intitule mode page. Il est utilisé dans le TO8, par l'intermédiaire du gate array correspondant.

Le rafraîchissement de ces mémoires dynamiques se fait par adressages successifs des 512 lignes.

Les 41256 ne sont pas sélectées (bus de données déconnecté) lorsque le signal CASN n'est pas actif. Ce dernier remplace avantageusement une commande de CHIP SELECT.

Organisation générale

La figure 18 montre deux plans mémoires distincts composés chacun de huit boîtiers 41256. Le plan supérieur représente les 256 Ko de mémoire vive résidant dans l'unité centrale, l'autre plan représente les 256 Ko de mémoire vive appartenant à l'unité d'extension.

Chaque boîtier reçoit en commun les commandes de RASN et RAMWN en provenance du gate array. CAS1N et CAS2N autorisent la sélection des plans mémoire, conformément au tableau suivant par l'intermédiaire des commandes communes de CASN.

$\overline{CAS1}$	$\overline{CAS2}$	Plan mémoire actif
0	1	Résident
1	0	Extension
1	1	Aucun

Un bus de données RAMS est commun en entrée-sortie à tous les boîtiers des deux plans mémoire. Ce bus est en relation avec le bus de données du 6809 par l'intermédiaire d'un buffer 74 LS 245 commandé par le signal TRANSN selon certaines modalités. (cf. chapitre II le 6809 E dans le TO8, page 95).

L'adressage de chaque boîtier, nécessairement multiplexé en 2 × 9 bits, est issu du gate array mode page. Cet adressage physique sur 18 bits (A0 à A17-256 Ko d'investigation) permet un découpage logique en pages de 16 Ko, selon une correspondance en espaces logiques de la carte mémoire limitée par principe à un adressage sur 16 bits (A0 à A15-64 Ko d'investigation). Ces espaces logiques sont alors à considérer comme des banques mémoires, ce qui, vu d'un logiciel, ne change rien quant à l'ancienne structure du type TO9, mais est totalement différent au niveau matériel.

Corrélativement à l'action du gate array, les trois espaces mémoire RAM définis dans la carte mémoire ont chacun une correspondance en numéro de page dans un des deux plans physiques selon:

Espace logique	Espace physique	
écran	page 0	plan résident
système	page 1	plan résident
données	pages 2 à 31	plan résident ou extension

Sélections et correspondances

Contrairement au TO9, la sélection des zones mémoire qui était faite par aiguillage de CAS et bits de PIA est réalisée, dans le TO8, par un processus de **transformation d'adresses**. L'adressage logique, provenant de la programmation sur 16 bits, est transformé en 18 bits.

Le tableau ci-dessous montre la relation entre les conditions physiques et les découpages logiques obtenus pour assurer la compabilité avec les anciens montages.

Conditions physiques			Découpage logique		
$\overline{CAS1}$	$\overline{CAS2}$	Zone physique	Page	Zone logique	Appellation
0	1	00000-01FFF	0	4000-5FFF	RAM écran B
0	1	02000-03FFF	0	4000-5FFF	RAM écran A
0	1	04000-07FFF	1	6000-9FFF	RAM système
0	1	08000-0BFFF	2	A000-DFFF	Banque 0
0	1	0C000-0FFFF	3	A000-DFFF	Banque 1
0	1	10000-13FFF	4	A000-DFFF	Banque 2
0	1	14000-17FFF	5	A000-DFFF	Banque 3
0	1	18000-1BFFF	6	A000-DFFF	Banque 4
0	1	1C000-1FFFF	7	A000-DFFF	Banque 5
-	-	- -	-	- -	-
0	1	3C000-3FFFF	15	A000-DFFF	Banque 13
1	0	00000-03FFF	16	A000-DFFF	Banque 14
-	-	- -	-	- -	-
1	0	3C000-3FFFF	31	A000-DFFF	Banque 29

La commutation de CAS1N et CAS2N est dépendante de la programmation du gate array mode page (voir page 105) dans lequel cinq bits représentent le numéro de page.

On remarquera, à la vue de ce tableau, une correspondance logique/physique beaucoup plus rationnelle que dans les montages précédents.

Ecriture et lecture des RAMS

Comme pour le TO9, deux cas de figure sont à considérer.

• E = 1, phase active du 6809:

Le microprocesseur peut communiquer en lecture ou écriture avec les RAMS à travers le 74 LS 245 par le bus de données RAM.
La commande RAMW = R/WN.

• E = 0, phase non active du 6809:

RAMW = 1, les mémoires sont en lecture automatique. Les lignes sont adressées en incrémentation constante par les compteurs du gate array.

Afin de permettre leur rafraîchissement en deux temps, soit en accès mode page, les colonnes sont doublement adressées, avec un écart constant qui représente une variation de 8 Ko. Ainsi:

Un premier accès a lieu dans la page physique 0 à une première adresse AD telle que:

• 00000 < AD < 01FFF donc dépendante de la mémoire écran B (couleur).

Un deuxième accès a lieu dans la même page physique 0 à une adresse telle que:

• 02000 < AD + 8 Ko < 03FFF donc dépendante de la mémoire écran A (points).

Toute cette organisation est définie à partir du gate array "mode page".

5. Le gate array mode page CF 74021

Ce nouveau circuit à réseau logique est une extension et une amélioration sensible du gate array principal utilisé dans le TO9. A lui seul, il assure en effet:

– La distribution des signaux nécessaires à l'interface vidéo
– Les décodages d'adresses
– Les huit modes d'affichages
– Une nouvelle gestion des mémoires vives, dont le "mode page"
– Un fonctionnement polyvalent de commutation, d'adaptation et de changement de caractéristiques pour divers systèmes.

Définition du mode page

Dans le TO9, le gate array système est conçu de telle sorte que lorsque le CPU travaille dans la mémoire écran, son accès est dirigé, soit vers la RAMA (mémoire points ou forme), soit vers la RAMB (mémoire couleur) en fonction du bit de forme. Les circuits d'exploitation automatique (automate) de la mémoire écran permettent, quant à eux, la lecture **simultanée** des deux RAMS. Cette lecture est indépendante du bit de forme, afin de pouvoir élaborer en temps réel l'intégralité des signaux de visualisation destinés au tube cathodique d'affichage.

Pour chaque groupe de huit pixels ou GPL affichés à l'écran, la circuiterie vidéo doit lire 16 bits en mémoire: 8 bits pour l'octet "RAMA" et 8 bits pour l'octet "RAMB". Jusqu'à maintenant, cette lecture devait se faire en une seule fois, sur 16 bits, ce qui imposait à la mémoire écran, d'être organisée en deux bancs de 8 bits, alors que tout le reste de la mémoire système est sur un octet seulement. Cette organisation "irrégulière" n'est pas optimale économiquement et n'est pas favorable à l'utilisation des boîtiers mémoire dynamiques N × 1 bits qui sont pourtant les plus répandus.

Afin de pallier cet inconvénient, le nouveau gate array dénommé "mode page" utilise un automate qui remplace l'accès 16 bits à la mémoire écran par deux accès successifs très rapides et qui totalisent la même durée que l'ancien accès 16 bits.

Cette technique est rendue possible grâce à un mode d'adressage particulier des mémoires dynamiques: le mode page. Dans ce type de fonctionnement, plusieurs

cases mémoires du composant peuvent être lues en séquence, à condition que ces cases appartiennent toutes à la même rangée ou ligne de la matrice interne.

Ainsi pour une adresse poids faible constante et un signal RASN à l'état bas, deux accès sont possibles pour deux adresses poids fort différentes en corrélation avec deux fronts descendants du signal CASN.

Cette nouvelle structure implique donc une organisation nouvelle de la mémoire écran pour que les octets "RAMA" et "RAMB" soient accessibles en mode page. Evidemment, vu du logiciel, l'agencement de l'écran doit rester compatible avec les systèmes précédents, c'est-à-dire que le bit de forme doit toujours permettre de rendre l'accès du CPU, soit à la mémoire "écran A" soit à la mémoire "écran B".

Gestion de la mémoire vive

Cette nouvelle version du gate permet de surcroît de gérer une quantité de RAM bien supérieure à celle implicitement contenue dans la carte mémoire. Ainsi est-il possible d'exploiter jusqu'à 512 Ko de mémoire vive par page de 16 Ko. Le numéro de page peut être choisi par programmation.

De même ces pages "physiques" peuvent être affectées à plusieurs espaces "logiques" du système, tels que l'espace "cartouche", l'espace "écran" et l'espace "données" (cf. gate array mode page dans le TO8, page 117).

D'un façon indépendante et de par une organisation très souple, un simple changement de programmation dans des registres rend possible l'adaptation du gate array mode page à diverses unités centrales (TO8, TO9+, MO6, MO5NR). De la même manière, divers types de mémoires dynamiques peuvent être câblés (4416, 4464, 4164, 41128, 41256 ou 44256).

Structure du circuit

La figure 19 schématise la structure interne du circuit à réseau logique prédiffusé. On y distingue:

– Le décodeur d'adresses sollicité par le bus d'adresses du microprocesseur A15 - A0.

– Le module des modes d'affichage avec ses deux registres programmables et son électronique en partie identique à ceux du gate array "mode d'affichage" dans le TO9. Conformément au mécanisme du "mode page", dans ce module le transcodeur est attaqué en deux accès de huit bits, en remplacement de l'ancien accès 16 bits RAMA RAMB. (Cf. analyse matérielle du TO9, p. 54.)

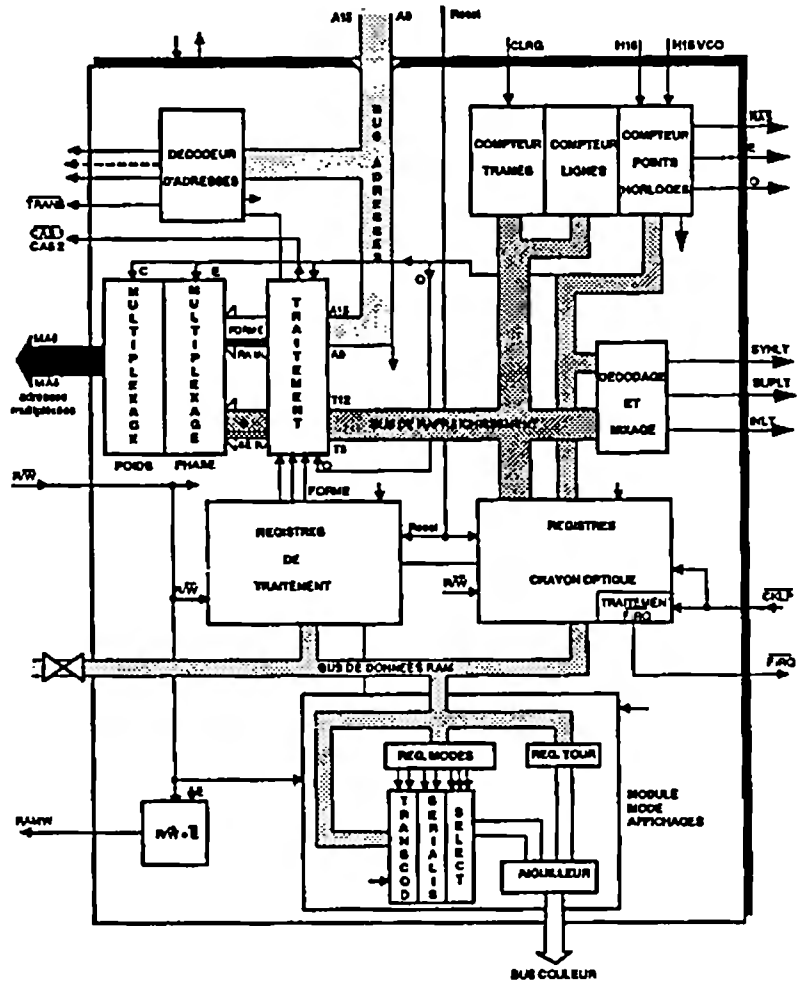


Figure 19. Synoptique du gate array mode page

– Le bus couleur en sortie du module.

– Un circuit de génération RAMW (en logique câblée dans le TO9), combinaison de R/WN et E, afin de forcer une lecture automatique des mémoires vives pendant la phase non active du CPU, selon l'équation :

$$RAMW = R/\overline{W} + \overline{E}$$

Comme pour le TO9, (cf. étude matérielle, p. 48) on reconnaît l'automate constitué:
– des compteurs points (avec C, RASN, E et Q),

– des compteurs ligne et trame, (pilotés par H16 et CLRG),
– et du bus de rafraîchissement T12 - T0 correspondant.

H16 VCO est une nouvelle commande de remise en phase (cf. incrustation).

– Les circuits de décodage et mixage délivrant les signaux vidéo SYNLT, SUPLT et INLT.

– Un système de registre pour le crayon optique quelque peu modifié dans son exploitation et permettant notamment le contrôle de FIRQN anciennement assuré par un PIA.

– Un double circuit de multiplexeurs commutant d'une façon classique les adresses CPU ou les adresses compteurs (phase du CPU en fonction de E); commutant en matriçage ligne et colonne (poids faible, poids fort en fonction de C); le tout, par l'intermédiaire du bus de neuf bits MA0-MA8 destiné aux mémoires vives.

En fait, un traitement ou aiguillage de signaux est effectué avant multiplexage. Cette action définit, entre autres, la génération des signaux CAS1N, CAS2N. Elle dépend de registres, dits de traitement, et programmables à souhait. C'est par là-même que résident l'originalié et la grande nouveauté du composant, voire du micro-ordinateur.

Traitement des signaux multiplexés

Les signaux multiplexés délivrés sur le bus MA8 - MA0 répondent au diagramme suivant:

		E=1 C=1	E=1 C=0	E=0 C=1	E=0 C=0	
MA0 - 7	⇒	A0	RA15	T0	BE1	BE1
MA1 - 8	⇒	A1	MU8	T1	TU8	TU8
MA2 - 9	⇒	A2	A9	T2	T9	T9
MA3 -10	⇒	A3	A10	T3	T10	T10
MA4 -11	⇒	A4	A11	T4	T11	T11
MA5 -12	⇒	A5	A12	T5	T12	T12
MA6 -13	⇒	A6	RA13	T6	Q=0	Q=1
MA7 -14	⇒	A7	RA14	T7	BE0	BE0
MA8 -15	⇒	MU16	RA17	T8	0	0
		ligne	colonne	ligne	col.1	col.2

On voit apparaître selon les commandes d'horloge interne :

Pour: – E = 1 (phase active du CPU)
– C = 1 (adresses RAM lignes)

A7-A0 = adresses poids faible du CPU; MU16 représentant soit A8, soit les bits de commutation de banques BC2 ou BD2.

Pour: – E = 1 (phase active du CPU)
– C = 0 (adresses RAM colonnes)

A9-A12 = adresses poids fort du CPU;
RA13 = soit A13, soit le bit de forme;
RA14 = soit A14, soit les commutations de banques BC0 ou BD0;
RA15 = les commutations de banques BC1 ou BD1;
MU8 = soit les bits de commutation de banques BC2 ou BD2, soit A8; MU8 = MU16 en commutation inverse;
RA17 = les commutations de banques BC3 ou BD3.

Pour: – E = 0 (phase non active du CPU)
– C = 1 (adresses RAM lignes)

T0-T8 = adresses poids faible des compteurs.

Pour: – E = 0 (phase non active du CPU)
– C = 0 (adresses RAM colonnes)

T9-T12 = adresses poids fort des compteurs;
TU8 = soit T8, soit "0";
BE0, BE1;
Q:
"0".

On notera que pour la condition E = 0, le système permet deux accès RAM, validés par deux fronts descendants de CASN (cf. gestion des mémoires vives) avec RASN à l'état 0. La différence d'adressage est représentée par l'état de Q qui, compte tenu du timing (quadrature avec E), délivre pendant la condition présente l'état 0 et l'état 1.

Q est en position de poids 13, ce qui représente une variation ou saut d'adresse de 2^{13} = 8 Ko, les autres bits restant inchangés. Ainsi, quelle que soit l'adresse pointée, il existe systématiquement deux accès dans deux pages de 8 Ko conjointes. Ces deux pages sont concrétisées par la RAMA et la RAMB.

Cette procédure réalise l'adressage automatique en mode page. On notera aussi que les signaux du type MU, TU, RA représentent des choix de commande imposés par programmation dans le registre de traitement afin de rendre compatible, comme nous l'avons précisé précédemment, le gate array avec le type de mémoires dynamiques et le micro-ordinateur choisis.

BC, BD, BE et le bit de forme, quant à eux, sont des états directement programmables dans les registres de traitement, pour les commutations de banque, voire de page.

Ainsi:
– BC3, BC2, BC1, BC0 fixent le choix d'une banque de 16 Ko parmi 16 banques, pour recouvrir l'espace "cartouche".

– BD3, BD2, BD1, BD0 fixent le choix d'une banque de 16 Ko parmi 16 banques pour recouvrir l'espace "données".

– BE1, BE0 fixent le choix d'une banque de 8 Ko parmi 4 banques, pour recouvrir l'espace "écran".

Les registres de traitement

En dehors de la gestion des mémoires vives, ces circuits offrent, en relation avec les horloges et les décodages d'adresses, des possibilités multiples d'adaptation et de changement de caractéristiques selon les machines à concevoir, sans oublier pour autant la compatibilité avec les anciennes versions.

Ces registres sont accessibles à des adresses bien particulières et par la commande du microprocesseur R/WN. Les adresses sont fondamentalement différentes selon que le composant travaille avec une unité centrale TO ou MO. La différence joue sur le digit hexadécimal de poids fort.

Ainsi: Digit de poids fort = A en version MO
Digit de poids fort = E en version TO

Description et programmation des registres accessibles en écriture

• Registre "système 1" - adresse A7E7/E7E7

Organisation:

D7 – bit de choix de l'utilisation du contrôleur de disque interne ou externe. En mode TO uniquement, ce bit a une influence sur les décodages d'adresses.

D7 = 0 ⇒ contrôleur interne
D7 = 1 ⇒ contrôleur externe

D6 – bit de gestion RAM dans l'espace cartouche.

D6 = 0 ⇒ mode compatible nanoréseau
D6 = 1 ⇒ gestion par registre interne"cartouche" A7E6/E7E6.

D5 – bit de standard d'affichage

D5 = 0 ⇒ 624 lignes (France)
D5 = 1 ⇒ 524 lignes (Export).

D4 - bit de gestion RAM dans l'espace "données".

D4 = 0 ⇒ gestion par bit de PIA (émulation)
D4 = 1 ⇒ gestion par registre interne: autorise l'écriture dans le registre "RAM données" en A7E5/E7E5.

D3 et D2 – bits de choix du type d'ordinateur.

D3 = 0 D2 = 0 ⇒ MO
D3 = 0 D2 = 1 ⇒ TO9
D3 = 1 D2 = 1 ⇒ TO

D1 et D0 – bits de choix de la RAM dynamique

D1 = 0 D0 = 0 ⇒ 256 K × 1 bit
⇒ 256 K × 4 bits
D1 = 1 D0 = 0 ⇒ 128 K × 1 bit
D1 = 1 D0 = 1 ⇒ 64 K × 4 bits

• Registre "système 2" - adresse A7DD/E7DD

Ce registre est une combinaison de l'électronique de traitement et du registre définissant la couleur du tour ou cadre dans le module d'affichage.

Organisation:

D7 et D6 – bits indiquant le numéro de page physique à afficher sur l'écran (de 0 à 3 en binaire naturel).

Avec D7 = BE1, D6 = BE0 } cf. diagramme précédent page 108

D5 – bit pour masquer la cartouche, utilisable uniquement en mode MO.

D5 = 0 ⇒ cartouche visible
D5 = 1 ⇒ cartouche masquée.

D4 – bit du BASIC à sélectionner, utilisable uniquement en mode MO.

D4 = 0 ⇒ BASIC 1
D4 = 1 ⇒ BASIC 128

D3, D2, D1, D0 – bits de la couleur du tour (cf. affichage).

• Registre "RAM données" - adresse A7E5/E7E5

Ce registre n'est accessible en écriture que si le bit D4 du registre "système 1" est écrit à 1.

Organisation:

D7 – bit d'autorisation d'accès au registre d'affichage en A7DC/E7DC en mode contrôleur de disque externe sélectionné (D7 de A7E7/E7E7 écrit à 1).

D7 = 0 ⇒ écriture autorisée
D7 = 1 ⇒ écriture inhibée.

Le rôle de ce bit est dû au fait d'un risque de conflit, à l'adresse A7DC/E7DC, avec un éventuel contrôleur externe de QDD qui décode lui aussi cet octet.

D6 = 0
D5 = 0

D4, D3, D2, D1, D0 – bits définissant le numéro de page RAM utilisée dans l'espace "RAM données" (de 0 à 31 en binaire naturel).

Avec :

D4 = Commutation de CASN

D4 = 0 ⇒ CAS1N valide
D4 = 1 ⇒ CAS2N valide

D3 = BD3
D2 = BD2
D1 = BD1
D0 = BD0
} cf. diagramme précédent, page 108

Ainsi, en mode gestion de l'espace "RAM données", D4-D0 donne le numéro de page physique de 16 Ko à affecter à l'espace logique.

• Registre "cartouche" adresse A7E6/E7E6

Organisation:

D7 = 0

D6 – bit de protection en écriture dans la page de RAM sélectionnée lorsque l'espace cartouche est recouvert par cette même page de RAM (D5 = 1).

D6 = 0 ⇒ écriture impossible
D6 = 1 ⇒ écriture autorisée.

D5 – bit de sélection de l'espace cartouche.

D5 = 0 ⇒ l'espace cartouche n'est pas recouvert par de la RAM.
D5 = 1 ⇒ l'espace cartouche est recouvert par une page de RAM dont le numéro est donné par D4-D0.

D4, D3, D2, D1, D0 – bits définissant le numéro de page RAM utilisée dans l'espace cartouche (de 0 à 31 en binaire naturel).

Avec :
D4 = Commutation de CAS.

D4 = 0 ⇒ CAS1N valide
D4 = 1 ⇒ CAS2N valide.

D3 = BC3
D2 = BC2
D1 = BC1
D0 = BC0
} cf. diagramme précédent, page 108

Ainsi, en mode de gestion de l'espace cartouche, D4-D0 donne le numéro de page physique de 16 Ko à affecter à l'espace logique correspondant.

• Registres "d'émulation"

Ces registres viennent remplacer, pour D4 du registre système 1 = 0, l'action dévolue aux PIA (6846, 6821) des unités centrales TO7/70, TO9, MO5, par la commutation du bit de forme et des banques. Réagissant aux même adresses, ils sont ainsi parfaitement transparents pour l'utilisateur. Ils assurent une compatibilité totale dans l'emploi des anciens logiciels.

• Registre de commutation "lecture traitement/crayon optique" adresse A7E4/E7E4.

Ce registre est un peu particulier et ne rentre pas dans la catégorie des registres susnommés. Il est en effet destiné selon l'écriture du bit D0 à aiguiller, pour des adresses semblables, la lecture de certains registres de traitement ou des registres crayon optique (latch).

Ainsi:
D0 = 0 ⇒ sélection des registres de traitement (lecture)
D0 = 1 ⇒ sélection des registres du crayon optique (lecture).

Description des registres accessibles en lecture pour D0 = 0 (A7E4/E7E4)

Excepté le registre "système 1", les registres de traitement, étudiés en écriture sont lisibles en entier ou partiellement, à des adresses différentes ou semblables.

• Registre "système 2" - adresse A7E4/E7E4

Il permet une relecture partielle du registre écrit en A7DD/E7DD.

Organisation:

D7 et D6 – bits indiquant le numéro de page physique affichée sur l'écran (de 0 à 3 en binaire naturel).

D5 – bit de masquage cartouche en mode MO.

D4 – bit de sélection du BASIC en mode MO.

Les bits D3, D2, D1, D0 sont à zéro.

• Registre "RAM données" - adresse A7E5/E7E5

Il permet la relecture du numéro de la page RAM, imposé en écriture, dans l'espace "données".

D7, D6, D5 sont à l'état "0".

• Registre "cartouche" - adresse A7E6/E7E6

C'est le seul registre permettant une relecture complète des bits positionnés pendant la phase d'écriture à la même adresse A7E6/E7E6. Il permet notamment la relecture du numéro de page RAM, imposé en écriture, dans l'espace cartouche.

Description des registres accessibles en lecture pour D0 = 1 (A7E4/E7E4)

Il s'agit de quatre registres semblables à ceux du gate array système dans le TO9.

• Registre "crayon optique 1" - adresse A7E4/E7E4

Organisation:

D7 = T12
D6 = T11
D5 = T10
D4 = T9
D3 = T8
D2 = T7
D1 = T6
D0 = T5

} Avec T12 - T5, 8 bits de poids fort du compteur trame.

• Registre "crayon optique 2" - adresse A7E5/E7E5

Organisation:

D7 = T4
D6 = T3
D5 = T2
D4 = T1
D3 = T0

} Avec T4-T0, 5 bits de poids faible du compteur trame.

D2 = E
D1 = H2
D0 = H4

} Avec E-H4, bits du compteur point.

• Registre "crayon optique 3" - adresse A7E6/E7E6

Organisation:

D7 – Bit significatif quand D6 = 0, c'est-à-dire, quand le spot est situé dans les bords droit ou gauche de l'écran.

D7 = 0 ⇒ spot situé dans la partie gauche du cadre.
D7 = 1 ⇒ spot situé dans la partie droite du cadre.

D6 – Bit latché de situation fenêtre cadre en ligne (INILN).

D6 = 0 ⇒ spot situé dans le cadre à gauche ou à droite.
D6 = 1 ⇒ spot situé dans la partie horizontale de la fenêtre de travail.

Cette valeur est latchée au moment de l'interruption de lecture crayon optique.

D5, D4, D3, D2, D1, D0 sont à l'état "0".

• Registre "crayon optique 4" - adresse A7E7 E7E7

Note: La lecture de ce registre est indépendante de l'état du bit D0 (A7E4/E7E4), ce qui le rend toujours accessible.

Organisation:

D7 – bit instantané de situation fenêtre cadre en trame (INITN)

D7 = 0 ⇒ spot situé dans le cadre en haut ou en bas.
D7 = 1 ⇒ spot situé dans la partie verticale de la fenêtre de travail.

D6 – Bit latché de situation fenêtre cadre en trame (INITN).

D6 = 0 ⇒ spot situé dans le cadre en haut ou en bas.
D6 = 1 ⇒ spot situé dans la partie verticale de la fenêtre de travail.

D5 – bit instantané de situation fenêtre cadre en ligne (INILN)

D5 = 0 ⇒ spot situé dans le cadre à gauche ou à droite.
D5 = 1 ⇒ spot situé dans la partie horizontale de la fenêtre de travail.

D4, D3, D2 sont à l'état "0".

D1 – bit de flag ou drapeau d'interruption de la demande FIRQN.

D1 = 0 ⇒ pas de demande.
D1 = 1 ⇒ une interruption a été générée.

D0 – bit de copie de D0 écrit en A7E4.E7E4 permettant de savoir quel type de registre est commuté en lecture. Ce bit est toujours accessible.

D0 = 0 ⇒ registres de traitement.
D0 = 1 ⇒ registres de crayon optique.

6. Le gate array "mode page" dans le TO8

Afin de procurer l'adaptation nécessaire à l'unité centrale, certains bits des registres de traitement "système 1" doivent être verrouillés dans un état bien particulier. La description suivante met en relief les bits nécéssairement "figés" pour la configuration TO8, par rapport aux bits "X" commutables selon les fonctions ou standard à réaliser.

Organisation du registre de traitement "système 1" en écriture - adresse E7E7

X	X	X	X	0	1	0	0
				type d'ordinateur		RAM 256 K × 1bit	

Ces conditions entraînent plus particulièrement:

MU16 = A8
MU8 = BC2,BD2
TU8 = 0

Les autres registres gardent la structure telle qu'elle est décrite dans le chapitre précédent, selon le répertoire suivant:

Adresse	R/W	Type de registre
E7E4	0	commutation ou système 2
E7E4	1	crayon optique 1
E7E5	0	RAM données
E7E5	1	crayon optique 2
E7E6	0	espace cartouche
E7E6	1	crayon optique 3
E7E7	0	système 1
E7E7	1	crayon optique 4
E7DD	0	système 2
E7DD	1	affichage

Diagramme des signaux multiplexés

Les conditions précédemment exposées déterminent le diagramme suivant:

A0	BC1,BD1	T0	BE1	BE1
A1	BC2,BD2	T1	0	0
A2	A9	T2	T9	T9
A3	A10	T3	T10	T10
A4	A11	T4	T11	T11
A5	A12	T5	T12	T12
A6	A13, forme	T6	Q = 0	Q = 1
A7	BC0, BD0	T7	BE0	BE0
A8	BC3, BD3	T8	0	0
Ligne	Colonne	Ligne	Col.1	Col.2

Les éléments de ce diagramme sont à considérer dans l'ordre des poids respectifs selon la forme:

Pour l'adressage CPU:

BC3 BC2 BC1 BC0 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
BD3 BD2 BD1 BD0 F

Adressage

Commutation des banques dans l'espace cartouche ou données

Commutation mémoires écran RAMA, RAMB

Il convient de rappeler que le bit D4 dans le registre "RAM données" et dans le registre "espace cartouche", de par sa position, joue indirectement le rôle d'un dix-neuvième bit d'adressage par le truchement de CAS1N et CAS2N.

De par ce fait, on peut donc considérer que le système est capable d'adresser un plan mémoire de 512 Ko constitué lui-même de deux plans mémoires de 41256 (résident et extension) et dont les adresses iraient de 00000 à 3FFFF pour D4 = 0, soit CAS1N = 0 et de 40000 à 7FFFF pour D4 = 1, soit CAS2N = 0.

Pour l'adressage de rafraîchissement ou compteurs:

0 0 BE1 BE0 Q T12 T11 T10 T9 T8 T7 T6 T5 T4 T3 T2 T1 T0

Adressage

Commutation des banques dans l'espace écran

Commutation mémoires écran RAMA, RAMB

On désigne ce type d'adressage comme provenant d'un automate.

Association entre adressage logique et adressage physique

Pour des raisons de compatibilité évidente, nous savons que l'ensemble du système, vu du logiciel, conduit à la détermination de quatre espaces d'adressage logique distincts:

l'espace "cartouche"
l'espace "écran" (page 0)
l'espace "système" (page 1 fixe)
l'espace "données" (pages des banques).

De par la programmation du gate array et la transformation de l'adressage résultant sur le bus multiplexé MA8-MA0, une page physique, c'est-à-dire 16 Ko d'un plan mémoire 41256, peut être affecté à un espace logique donné. Le comportement individuel de chacun des espaces logiques est spécifié dans ce qui suit (cf. figure 20 page suivante).

Espace "cartouche"

Cet espace situé entre 0000 et 3FFF contient normalement de la ROM sous forme de mémoire interne (BASIC 1, BASIC 512) ou de mémoire externe (cartouche). Il est néanmoins possible d'affecter une page de mémoire vive à cet espace logique (recouvrement) en mettant le bit D5 du registre gate array (E7E6) à l'état 1. Dans ces conditions, CSCRTN est dévalidé et les 41256 reçoivent CAS1N ou CAS2N. (cf. gestion de la mémoire morte, page 96).

La correspondance entre l'adressage CPU et l'adressage multiplexé est immédiate pour les bits A0 à A13. Elle se présente sous la forme suivante:

Adressage logique CPU (16 bits)

0 0 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0

Adressage physique transformé (18 bits)

BC3 BC2 BC1 BC0 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0

L'adressage transformé permet un recouvrement par une des 16 pages d'adresses de 00000 à 3FFFF pour CAS1N = 0 du plan mémoire résident et, par 16 pages d'adresses de 00000 à 3FFFF pour CAS2N = 0, du plan mémoire extension.

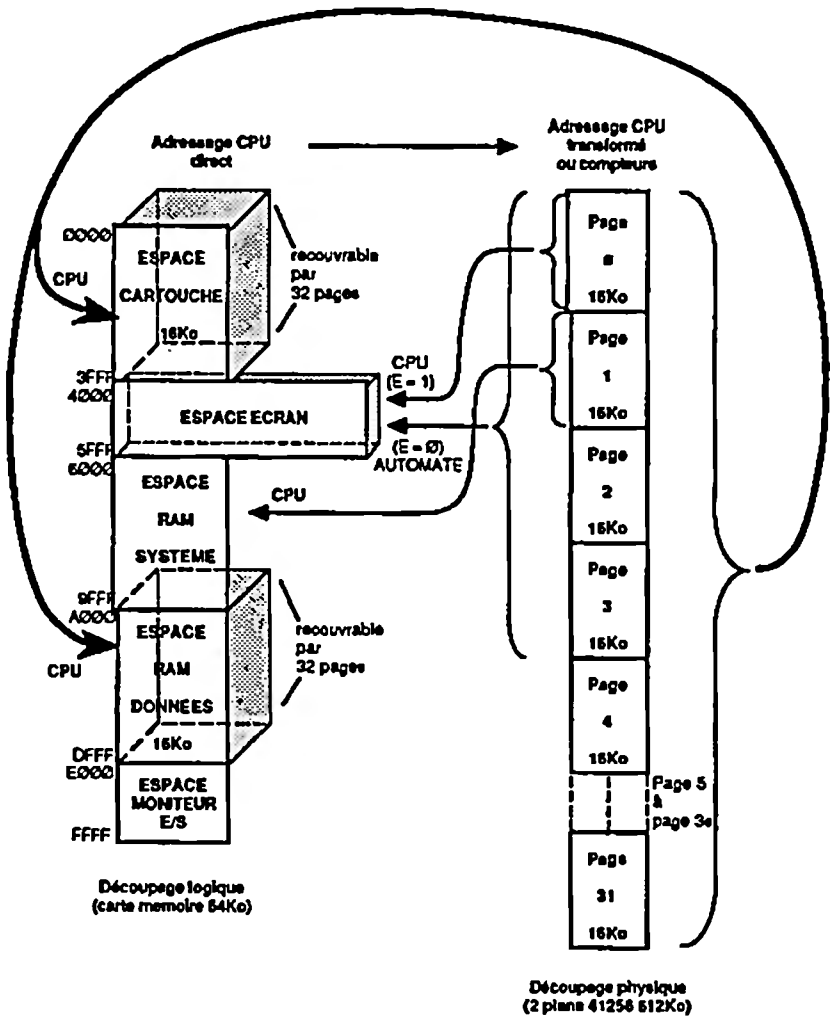


Figure 20. Association des espaces logiques/physiques

Espace "écran"

Cet espace, situé entre 4000 et 5FFF, contient deux banques de 8 Ko (RAMA ou mémoire point, RAMB ou mémoire couleur) utilisées par l'automate pour synthétiser l'image affichée. Par défaut, c'est la page 0 qui est affichée, mais il est possible de demander à l'automate d'afficher les pages 1, 2 ou 3.

Le principe est le suivant:

– Après un RESET, une mise sous tension ou si les bits D7, D6 du registre situé en E7DD sont tous les deux à 0, c'est la page physique 0 qui est affichée à l'écran. La façon dont les données sont interprétées par l'automate d'affichage dépend alors du mode choisi.

– En reprogrammant les bits D7 et D6 de E7DD, on peut afficher les pages physiques 1, 2 ou 3. Si on demande l'affichage de la page 1, c'est le contenu de la RAM "système" qui apparaîtra à l'écran. Ce contenu ayant peu d'intérêt, ce sont surtout les pages 2 et 3 qui seront utilisables.

– Lorsque le CPU accède à l'espace logique d'écran, c'est toujours la page physique 0 qui est adressée, même si l'automate d'affichage utilise une autre page. Dans l'espace logique d'écran, le CPU utilise le bit "FORME" pour travailler dans la mémoire RAMA (point) ou RAMB (couleur), de par la programmation en émulation du bit D0 en E7C3.

– Si le CPU veut accéder aux pages 2 et 3 en les considérant comme des écrans, il doit les affecter à son espace RAM "données" afin de pouvoir les lire ou les écrire. Dans ce cas, on doit considérer que la mémoire RAMA (point) se trouve aux adresses hautes de la page, tandis que les adresses basses contiennent les informations RAMB (couleurs); en effet, le bit FORME traditionnel est inopérant dans l'espace "données".

– La page affichée par l'automate peut être en même temps affectée à l'espace "données", voire à l'espace "cartouche". De par le principe énoncé, si on veut par exemple afficher la page 2, dans laquelle le CPU puisse faire une mise à jour, il faudra nécessairement affecter la page 2 à l'espace RAM "données" qui sera considérée alors comme la nouvelle mémoire écran; et demander à l'automate d'afficher la page 2, en programmant les bits D7 D6 en E7DD respectivement à 1 et 0.

Espace "système"

Cet espace logique situé entre 6000 et 9FFF contient les informations vitales à l'unité centrale, telle que la page 0 du moniteur. Cet espace n'est pas reconfigurable, on dit qu'il est fixe. C'est la page physique 1 du plan mémoire 41256 qui y est constamment affectée. Aucun registre ne permet de reprogrammer cet espace.

Espace "données"

Cet espace logique est situé entre A000 et DFFF. Sur TO9 et sur TO7/70 cet espace était géré en "bank switch" à l'aide des bits de PIA qui permettaient de choisir une page physique de 16 Ko parmi six. Toujours pour des raisons de

compatibilité, cette technique est ici totalement émulée par le gate array mode page. L'écriture et la lecture des cinq bits de PIA se font identiquement. Pour obtenir cette émulation, il faut que le bit D4 du registre E7E7 soit mis à 0. Ceci est obtenu par défaut au RESET et à la mise sous tension.

Nous savons qu'un deuxième mode est disponible en mettant à 1 le bit D4 de E7E7. Dans ce mode, il suffit d'écrire le numéro (de 0 à 31) de la page physique souhaitée, par les bits D4-D0 du registre E7E5, pour que la page concernée soit affectée à l'espace "données". Ainsi la gestion de la mémoire s'en trouve simplifiée puisqu'une simple écriture de numéro de page suffit à effectuer le recouvrement. La correspondance entre l'adressage CPU et l'adressage multiplexé est immédiate pour les bits A0 à A13. Elle se présente sous la forme suivante:

Adressage logique CPU (16 bits)

1 0
1 0
1 1 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
1 1

Adressage physique transformé (13 bits)

BD3 BD2 BD1 BD0 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0

Comme pour l'espace cartouche, l'adressage transformé permet le recouvrement des deux plans mémoire de 256 + 256 = 512 Ko par l'intermédiaire du CAS1N et CAS2N en relation avec le bit D4. Il faut noter cependant que de par la transposition schématisée ci-avant, il existe une distorsion entre les deux adressages qui se manifeste dans le champ A15-A12. Ainsi, en prenant par exemple un recouvrement par la page 2:

BD3 = 0 BD2 = 0 BD1 = 1 BD0 = 0,

la transposition devient:

Adressage logique					Adressage physique				
A15	A14	A13	A12	...	BD1	BD0	A13	A12	...
1	0	1	0	...	1	0	1	0	...
1	0	1	1	...	1	0	1	1	...
1	1	0	0	...	1	0	0	0	...
1	1	0	1	...	1	0	0	1	...

On traduit ce phénomène par un découpage en tranches de 4 Ko, exprimé par le tableau suivant, selon la valeur de A13, A12:

Adresse logique	Adresse physique
A000-AFFF	3ème tranche de 4 Ko
B000-BFFF	4ème tranche de 4 Ko
C000-CFFF	1ère tranche de 4 Ko
D000-DFFF	2ème tranche de 4 Ko

Cette transposition n'a pas d'importance tant que l'on ne transfère pas d'information de l'espace "données" vers les autres espaces. Si on opère des transferts, il faut alors tenir compte des correspondances.

Par déduction,la correspondance entre espace "cartouche" et espace "données" est la suivante:

Espace "données"	Espace "cartouche"
A000-AFFF	2000-2FFF
B000-BFFF	3000-3FFF
C000-CFFF	0000-0FFF
D000-DFFF	1000-1FFF

De même, la correspondance entre espace "données" et espace "écran" est telle que:

Espace "données"	Espace "écran"
A000-BFFF	RAMB
C000-DFFF	RAMA

La correspondance entre espace "cartouche" et espace "écran" est donc quant à elle:

Espace "cartouche"	Espace "écran"
0000-1FFF	RAMA
2000-3FFF	RAMB

On notera pour terminer que les six pages accessibles par bits de PIA, selon l'ancien mécanisme, correspondent aux pages 2 à 7 selon le nouveau système.

On en déduit:

Ancien système banque	Nouveau système page
0	2
1	3
2	4
3	5
4	6
5	7

Au RESET, la gestion par bits de PIA étant prise par défaut, c'est la page physique 2 qui est affectée à l'espace RAM "données".

Gestion de l'affichage

La procédure d'affichage sur l'écran est sensiblement identique à celle du TO9 (cf. étude matérielle du TO9, page 54). Le gate array mode page intègre la totalité de la circuiterie vidéo capable de gérer les différents modes d'affichage connus sur le TO9. Il est prévu pour piloter directement une palette externe.

Selon les modes d'affichage, les couleurs imposées en configuration de base diffèrent de celles connues sur le TO9. Le bit S (saturation) est notamment remplacé par le bit P (pastel).

En dehors du bit D5 du registre "système 1" en E7E7 qui permet de définir le standard 624 lignes (système européen) ou 524 lignes (système NTSC), un registre spécial appelé registre "affichage", situé en E7DC, contient les bits de programmation essentiels de l'affichage.

Ce registre d'affichage est identique à celui du TO9 (cf. système de visualisation du TO9) et, de la même manière, permet de décider:

- du mode d'affichage
- de la fréquence pixel
- de l'organisation relative des bits à l'intérieur des données vidéo extraites de la mémoire.

Cette procédure est récapitulée par le tableau suivant:

Mode	D7 0	D6 T1	D5 T0	D4 Φ1	D3 Φ0	D2 C	D1 B	D0 A
TO7	0	0	0	0	0	0	0	0
bit map 4	0	0	1	0	0	0	0	1
bit map 4 spécial	0	1	0	0	0	0	0	1
80 colonnes	0	0	1	0	1	0	1	0
bit map 16	0	1	1	1	1	0	1	1
page 1	0	0	1	0	0	1	0	0
page 2	0	0	1	0	0	1	0	1
surimpression	0	0	1	0	0	1	1	0
triple surimpression	0	0	1	1	1	1	1	1
		organ.		fréq. pixel		mode		

La consultation de ce tableau permet de voir qu'il existe un neuvième mode d'affichage non utilisé sur le TO9, **le mode bit map 4 couleurs spécial.** Le schéma de codage et transcodage en RAM est le suivant:

Mode bit map 4 (rappel)

RAMA								RAMB							
C7	C6	C5	C4	C3	C2	C1	C0	C7	C6	C5	C4	C3	C2	C1	C0
définition du rouge R								définition du vert V							

Après transcodage ==> mode bit map 4 spécial:

C7	C7	C6	C6	C5	C5	C4	C4	C3	C3	C2	C2	C1	C1	C0	C0
R	V	R	V	R	V	R	V	R	V	R	V	R	V	R	V

Dans ce type de codage, un point est représenté par deux bits consécutifs du même octet. Cette organisation présente un intérêt pour certaines routines graphiques; toutefois, il faut noter qu'elle n'est pas employée par les routines du moniteur.

Gestion des couleurs du cadre

La couleur du tour de l'écran est programmable directement par le registre "système 2" en E7DD, par les bits D3 à D0, selon une organisation semblable à celle du TO9, exception faite pour le bit S (saturation) qui devient P (pastel).

D3	D2	D1	D0
P	B	V	R

Les décodages d'adresses

Le gate array mode page génère directement un certain nombre de signaux destinés à sélectionner les différents ROMS et périphériques. Ce sont:

CSMN = sélection de l'espace moniteur
CSCRTN = sélection de l'espace cartouche
CS32 = sélection de 32 octets (FFD0-FFEF)
E7CXN = sélection de la zone des 6821 et 6846
CSPALN = sélection de la palette
EXXXN = sélection périphériques externes
DISN = sélection du contrôleur disque (E7D0 à E7D9)

Les signaux CS32, E7CXN, CSPALN sont décodés par une simple logique combinatoire et sont toujours valides. Les autres signaux dépendent de la programmation de certains registres et leur comportement est détaillé ci-après.

Sélection de l'espace moniteur CSMN

Le moniteur système du TO8 est situé de l'adresse E800 à FFFF. L'espace utilisé par le moniteur du drive (floppy ou QDD) est placé de E000 à E7AF. Les deux moniteurs faisant partie intégrante d'une même ROM, c'est le même signal qui est validé sur l'ensemble :

[E000 - E7AF] U [E800 - EFFF].

Par ailleurs, il est possible en programmant le bit D7 du registre "système 1" en E7E7 à l'état 1, d'installer un contrôleur de mémoire de masse externe. Dans ce cas de figure, CSMN n'est plus validé de E000 à E7AF.

Il est à signaler que l'espace moniteur contient un "trou" de 32 octets situé en FFD0 et FFEF qui sont réservés aux décodages d'extensions futures (rôle de CS32 en relation avec le connecteur extension). Dans cet espace, CSMN = 1.

Sélection de l'espace cartouche CSCRTN

L'espace logique réservé aux logiciels externes (cartouche) ou interne s'étend de 0000 à 3FFF. Il peut être recouvert par de la RAM en positionnant le bit D5 du registre "espace cartouche" à l'état 1. Dans ce cas, CSCRTN qui est normalement validé passe à l'état 1, supprimant, par là-même, toute communication avec les ROMS.

Sélection de la zone des périphériques externes EXXXN

Le signal EXXXN sort du gate array "mode page" pour aller vers les connecteurs d'extensions de l'unité centrale où il est utilisé par les périphériques pour leur décodage.

Il permet, entre autre, d'effectuer un prédécodage d'adresses dans le cas où l'utilisateur désire se fabriquer sa propre extension. Ce dernier devra alors impérativement loger son interface dans l'espace mémoire E7B0 à E7BF.

Comme dans le TO8, un bon nombre d'adresses correspondant à la zone EXXX sont déjà exploitées, il faut donc que le signal correspondant soit non valide quand il y a risque de conflit entre un périphérique extérieur et l'unité centrale.

Les possibilités de conflit diffèrent selon l'utilisation ou non d'un contrôleur externe.

En utilisation normale (D7 du registre "système 1" = 0), EXXXN est invalidé sur l'ensemble:

[E000 - E7AF] U [E7D0 - E7D9].

En utilisation d'un contrôleur externe (D7 = 1), EXXXN reste valide à ces endroits.

De même, une souplesse de programmation est laissée pour l'octet E7DC qui peut être à la fois employé en temps que registre d'affichage ou par un contrôleur externe de QDD.

Si le bit D7 du registre "données" en E7E5 est écrit à 0, alors EXXXN n'est pas valide à l'adresse E7DC; sinon EXXXN reste valide à l'adresse E7DC mais on ne peut plus accéder au registre d'affichage.

L'ensemble des adresses [E7DA - E7DB] U [E7DD - E7DF] invalide toujours le signal EXXXN car elles correspondent à la palette et certains registres protégés.

Sélection du contrôleur du DRIVE DISN

Ce signal n'est valide sur [E7D0 - E7D9] que si le bit D7 du registre "système 1" (E7E7) = 0, déterminant alors l'utilisation du contrôleur interne.

Tableau récapitulatif

Avec D77 bit du registre "système 1" en E7E7 pour le contrôleur externe.

Avec D75 bit du registre "RAM données" en E7E5 pour le choix d'un QDD.

	D77 = 0			D77 = 1			
	D75 = 0		D75 = 1				
Espaces	CSMN	EXXXN	DISN	CSMN	EXXXN	DS1N	EXXXN.
E000-E7AF	0	1	1	1	0	1	0
E7B0-E7CF	1	0	1	1	0	1	0
E7D0-E7D9	1	1	0	1	0	1	0
E7DA-E7DB	1	1	1	1	1	1	1
E7DC	1	1	1	1	1	1	0
E7DD-E7DF	1	1	1	1	1	1	1
E7E0-E7FF	1	0	1	1	0	1	0
E800-EFFF	0	1	1	0	1	1	1
							QDD
	contrôleur interne			contrôleur externe			

Gestion du crayon optique

Le fonctionnement de l'interrupteur du crayon optique est assuré de la même manière que sur le TO9 (cf. gestion du crayon optique, page 82).

Pour la gestion du photo-transistor, via un circuit à miroir de courant, le signal CKLPN est récupéré par le gate array mode page qui intègre la demande d'interruption FIRQN établie anciennement par le 6821 dans le TO9. En dehors de cette particularité, le fonctionnement reste pratiquement semblable et le gate array mode page offre, comme son successeur, la possibilité de faire des mesures avec la résolution du point.

Nous savons que quatre registres permettent de lire les informations afférentes à une mesure light-pen; ce sont :

en E7E4 - le registre crayon optique 1
en E7E5 - le registre crayon optique 2
en E7E6 - le registre crayon optique 3
en E7E7 - le registre crayon optique 4

En écrivant le bit D0 de E7E4 à 1, on se prépare à pouvoir lire ces registres et, par la même occasion, on autorise le passage du signal arrivant du crayon optique lui-même vers la sortie ITLP du gate array reliée à l'entrée FIRQN du CPU. On "arme" donc la routine light-pen en rendant le CPU interruptible par les signaux de mesure en provenance du crayon optique. L'arrivée d'une impulsion de mesure CKLPN provoque une FIRQN sur le CPU et latche le contenu du compteur 16 bits associé. La valeur du compteur désigne alors le pixel "vu" par le crayon optique. Le CPU peut alors lire le contenu de ce compteur par les registres E7E4 et E7E5.

– E7E4 donne les 8 bits de poids fort:

T12 T11 T10 T9 T8 T7 T6 T5

– E7E5 donne les 8 bits de poids faible:

T4 T3 T2 T1 T0 E H2 H4

Les 3 bits E, H2, H4 issus d'un comptage à 8 MHz donnent la résolution du pixel dans les modes 320 × 200 et d'un pixel sur deux en mode 80 colonnes. D'autres informations complétant la lecture de ces compteurs sont disponibles dans les deux autre registres (cf. Description des registres accessibles en lecture pour D = 0, page 114).

Au moment de l'arrivée de CLKLPN, un drapeau d'interruption est levé dans le bit D1 du registre E7E7. Ceci permet de vérifier que l'interruption FIRQN a bien été causée par le crayon optique.

La lecture des poids faibles du compteur light-pen (E7E5) a pour effet de relâcher la ligne FIRQN et d'abaisser le drapeau d'interruption dans le registre E7E7. Dès que cette lecture est effectuée, le gate array est donc prêt à effectuer une autre mesure.

L'organisation du gate array mode page permet donc d'implémenter une routine de mesure du crayon optique similaire à celle qui existe sur le TO9 (cf. étude matérielle du TO9, gestion du crayon optique, page 82).

7. Chaîne de visualisation

Le gate array "mode page" génère les informations R, V, B, P et HP nécessaires au circuit de palette IGV EF 9369 d'une façon parfaitement identique au TO9 (cf. Le système de visualisation du TO9, page 52). Ce circuit remplit bien évidemment les mêmes fonctions (choix de 16 teintes parmi 4096, choix de la couleur d'incrus-tation) et se programme de la même manière.

Les trois sorties analogiques R, V, B attaquent, d'une façon toujours identique au TO9, la prise péritel à travers un circuit d'adaptation et un mélangeur recevant le signal SYNLT pour reconstituer une vidéo composite.

Les trois sorties sont dérivées vers un adaptateur pour un éventuel modulateur PAL (version EXPORT).

Le circuit d'incrustation, bien que semblable à celui du TO9 dans son principe (cf. Circuit d'incrustation du TO9, page 71), diffère par sa conception. En effet, le gate array "mode page" possède une entrée spéciale (16MHz VCo) de remise en phase, ce qui, en demande d'incrustation, ne nécessite pas une décommutation de l'horloge mère.

La figure 21 concrétise le fonctionnement général du dispositif.

Ainsi en fonctionnement normal (après mise sous tension ou RESET) le bit CB2 du PIA système est à l'état 1. Le transistor T14 est saturé et le transistor TO6 est bloqué. Le 12 V apparaît en commande de commutation lente et force par la diode D4 et la conduction du transistor TO5 la commande de commutation rapide. Le bit marqueur en provenance du circuit de palette est inopérant, la diode D7 étant bloquée.

En fonctionnement incrusté, après mise à l'état 0 de CB2 par programmation, le transistor T14 est bloqué, entraînant la saturation du transistor TO6. La commande de commutation lente devient inactive (0 V). Le bit marqueur, protégé par la diode D4 constamment bloquée, devient maître d'œuvre pour envoyer la commande de commutation rapide à travers TO5, selon la procédure connue sur le TO9:

couleur hors incrustation ==> $\overline{M} = 1$ ==> commutation rapide

couleur d'incrustation ==> $\overline{M} = 0$ ==> pas de commutation rapide

Le boîtier d'incrustation branché, le gate array reçoit la commande de remise en phase ligne par l'intermédiaire du 16 MHz VCo. Il reçoit la commande de remise à l'heure trame par l'intermédiaire de CLRG.

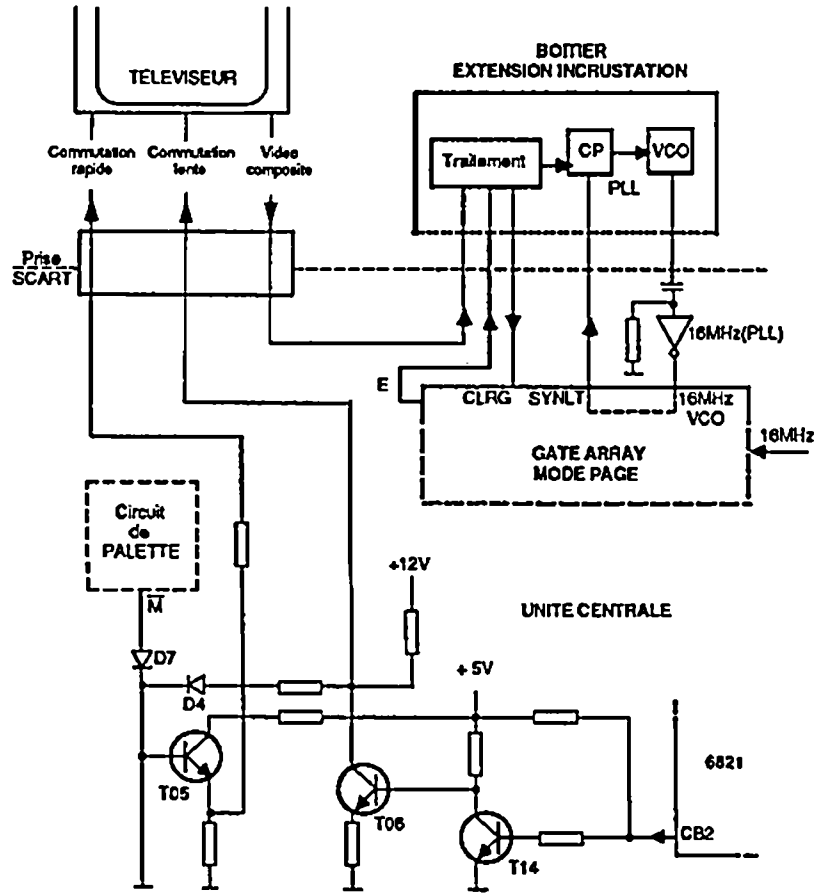


Figure 21. Synoptique du fonctionnement de l'incrustation dans le TO8

8. Les interfaces

Le TO8 comprend quatre circuits d'interface pour gérer ses divers périphériques (un 6846, deux 6821, un 6804). Certains bits restent compatibles pour le TO9, ce qui est mis en évidence dans la description suivante.

L'interface 6846

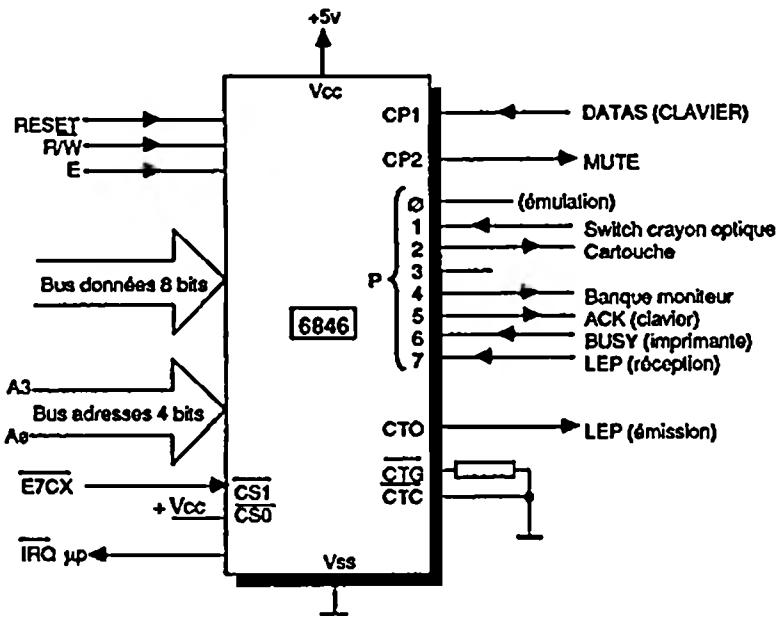


Figure 22. Le 6846 dans le TO8

Partie ROM

Comme pour le TO9, la partie mémoire morte de ce boîtier complexe n'est pas utilisée (adressage sur 4 bits de A0 à A3).

Partie PIA

Le port 8 bits a, par initialisation, trois lignes P2, P4, P5 configurées en sortie et trois autres lignes P1, P6, P7 en entrée. Le bit P0, bien que configuré en sortie, n'est pas utilisé matériellement (broche non connectée). En fait, l'action de ce bit est émulée dans le gate array "mode page" (cf. Le gate array mode page CF 7021, page 105) pour la commutation du bit de forme.

– Le bit P1, comme pour le TO9, assure la lecture de l'état de l'interrupteur du crayon optique (0 ==> interrupteur ouvert, 1 ==> interrupteur fermé).

– Le bit P2, incompatible sur le TO9, commande la commutation cartouche/logiciels internes (cf. Sélection entre logiciels résidents et cartouche, page 98).

– Le bit P3 n'est pas utilisé (ancienne commande LED clavier TO7, TO7/70).

– Le bit P4, incompatible sur le TO9, détermine la sélection de page moniteur (cf. Sélection d'une page moniteur, page 98) avec:

0 ==> partie basse.
1 ==> partie haute.

– Le bit P5, incompatible sur le TO9, envoie le signal ACK (acknoledge-accusé de réception), en retour d'une communication du clavier (cf. Le 6804, page 138).

– Le bit P6, incompatible sur le TO9, reçoit l'information BUSY (indicateur d'occupation) de l'imprimante CENTRONICS avec:

0 ==> imprimante occupée
1 ==> imprimante disponible

– Le bit P7 réceptionne, comme pour le TO9, les informations numériques décodées en lecture du LEP.

Les lignes de contrôles, incompatibles avec celles du TO9, assurent pour:

– CP1: La réception des données en transmission série, provenant du clavier via le 6804. Chaque bit est codé selon une temporisation récupérée en demande IRQN par le 6809 E (cf. fonctionnement du 6804).

– CP2: initialisée en sortie, envoie la commande "MUTE" procurant un blocage du son (action sur un transistor du mélangeur) lorsque l'utilisateur manipule la souris:

0 ==> passage du son
1 ==> blocage du son

Partie TIMER

D'une façon identique au TO9, la sortie CT0 génère l'écriture cassette.

Adresses des registres internes

E7C0 - registre d'état composite
E7C1 - registre de contrôle périphérique
E7C2 - registre de direction de données
E7C3 - registre de données périphériques
E7C4 - registre d'état composite
E7C5 - registre contrôle temporisation
E7C6 - registre temporisation

Le 6821 système

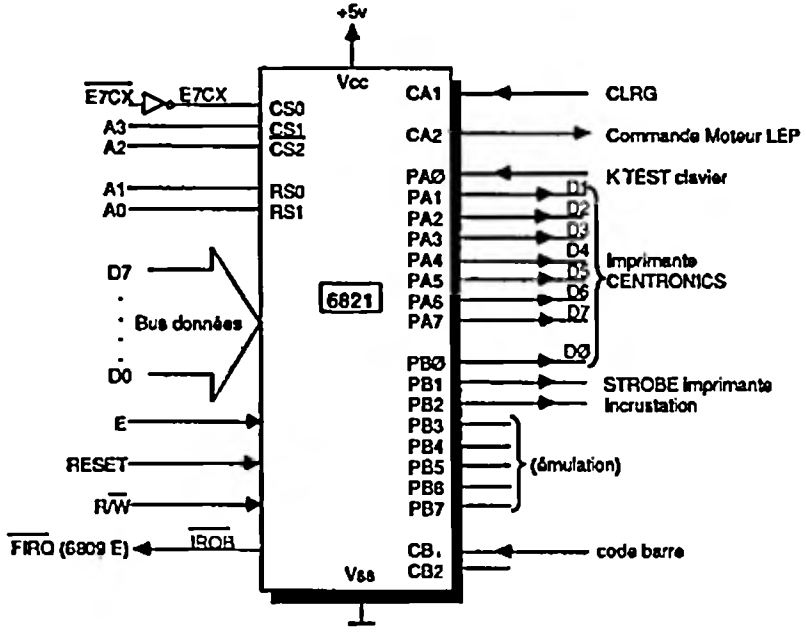


Figure 23. Le 6821 "système" dans le TO8

A l'exception de CB1 recevant par la broche 5 du connecteur crayon optique, une éventuelle demande en code barre, et des bits PB7, PB6, PB5, PB4, PB3 de commutation banque RAM sur le TO9, dont l'action est ici parfaitement émulée par le gate array "mode page" (les broches sont "en l'air"), le composant joue le même rôle que dans l'unité centrale du TO9 (cf. Utilisation du 6821 dans le TO9, page 75).

Adresses des registres internes

E7C8 - registre de direction de données ou registre de données partie A
E7C9 - registre de direction de données ou registre de données partie B
E7CA - registre de contrôle partie A
E7CB - registre de contrôle partie B.

Le 6821 jeux et musique

Ce boîtier a, de par l'initialisation, toutes ses lignes en entrée. La quasi-totalité des lignes assume l'exploitation des manettes de jeux ou de la souris. Lorsque l'unité centrale doit émettre un message sonore, six bits du port B sont alors commutés en sortie pour attaquer, après bufferisation, un convertisseur numérique analogique du type CNA R/2R dont la sortie est reliée au mélangeur recevant par ailleurs une éventuelle information audio, après lecture du LEP.

Description des broches

– Port A

Quatre lignes sont consacrées à la manette de jeux "0", ou à la souris, par l'intermédiaire du connecteur B12 (prise avant) avec:

PA0	contact nord	ou gachette 1
PA1	contact sud	ou gachette 2
PA2	contact ouest	ou XB
PA3	contact est	ou YB

Les quatre autres lignes sont consacrées à la manette de jeux "1" par l'intermédiaire du connecteur B13 (prise arrière) avec:

PA4	contact nord
PA5	contact sud
PA6	contact ouest
PA7	contact est

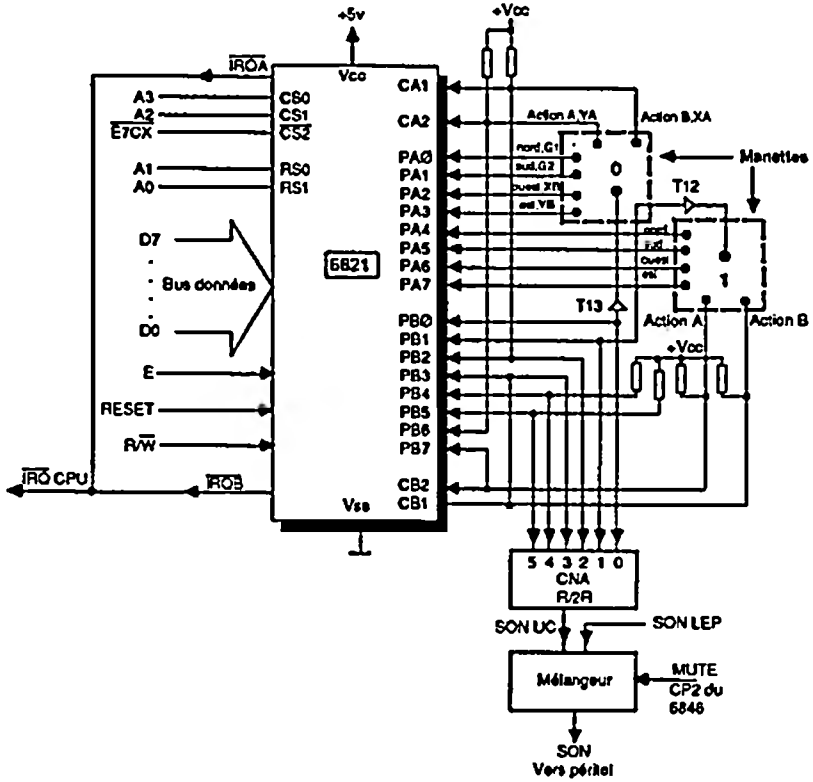


Figure 24. Le 6821 "musique et jeux" dans le TO8

– Port B

Chaque ligne a un rôle complexe:

PB0 commun bufferisé pour la manette de jeux "0" ou bit 0 configuré en sortie pour le CNA.

PB1 commun bufferisé pour la manette de jeux "1" ou bit 1 configuré en sortie pour le CNA.

PB2 bouton d'action B de la manette de jeux "0" ou XA de la souris ou bit 2 configuré en sortie pour le CNA.

PB3 bouton d'action B de la manette de jeux "1" ou bit 3 configuré en sortie par le CNA.

PB4 bit 4 configuré en sortie pour le CNA.

PB5 bit 5 configuré en sortie pour le CNA.

PB6 bouton d'action A de la manette de jeux "0" ou YA de la souris.

PB7 bouton d'action A de la manette de jeux "1".

Les lignes de contrôle CA1, CA2, CB1, CB2 peuvent être utilisées conjointement en demande IRQN pour les boutons d'action ou la souris selon:

CA1 bouton d'action B pour la manette "0"

CA2 bouton d'action A pour la manette "0"

CB2 bouton d'action A pour la manette "1"

CB1 bouton d'action B pour la manette "1".

De par l'emploi des mêmes connections pour l'élaboration du son et pour l'utilisation de la souris, il est nécessaire, pour ne pas être gêné par un bruit parasite à chaque manipulation, de bloquer la sortie son. Cela est réalisé par le bit CP2 du PIA 6846 (0 sortie son validée - 1 sortie son invalidée) qui vient agir sur le mélangeur.

Adresses des registres internes

E7CC - registre de direction de données ou registre de données partie A

E7CD - registre de direction de données ou registre de données partie B

E7CE - registre de contrôle partie A

E7CF - registre de contrôle partie B.

Le 6804

Le MONOCHIP 6804 a pour tâche l'interfaçage et le traitement du clavier en relation avec le 6846 et le 6821 système.

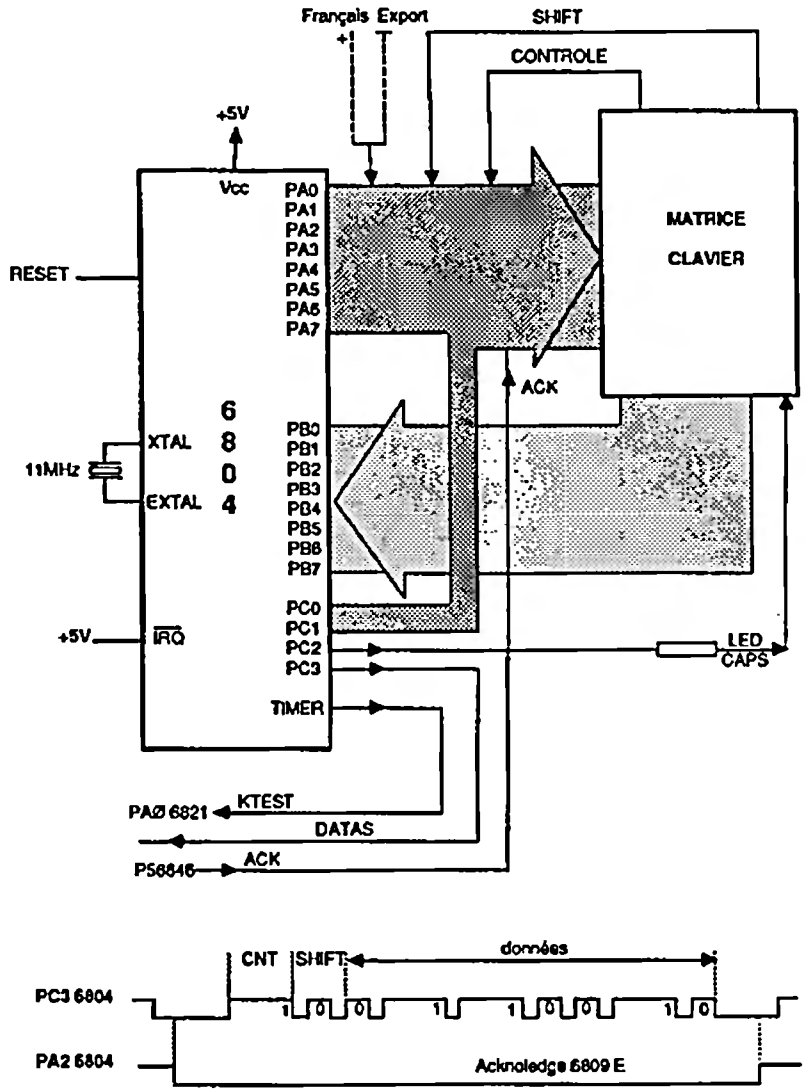


Figure 25. Gestion du clavier dans le TO8

Interfaçage du clavier

La figure 25 montre les interconnexions entre la matrice 10 × 8 du clavier et les ports A, B et C du 6804. Ce dernier est piloté par un quartz à la fréquence de 11 MHz.

Fonctionnement

Comme pour le TO9, c'est le clavier qui indique au 6809 E qu'une touche vient d'être appuyée. La transmission est réalisée par un dialogue entre les deux microprocesseurs via trois bits de PIA:

- CP1 du 6846, pour les données à transmettre
- P5 du 6846, pour la reconnaissance d'une touche du clavier (ACK),
- PA0 du 6821 système pour la reconnaissance du périphérique (KTEST).

• Emission d'une donnée par le microprocesseur "clavier":

Les données émises par le microprocesseur "clavier" se composent de 9 bits:

1er bit:	Touche CNT enfoncée	--> 1
	Touche CNT libre	--> 0
2ème bit:	Touche SHIFT enfoncée ou CAPS LOCK actionné	--> 1
	Touche libre et CAPS LOCK non actionné	--> 0

7 bits suivants: Numéro de la touche de 0 à 79 (contrairement au TO9 et TO9 +, ce n'est pas le code ASCII de la touche qui est envoyé, mais seulement le numéro de la touche, le trancodage numéro --> code ASCII étant à la charge du 6809 E).

Corrélativement avec KTEST, pour prévenir le 6809 E qu'il va recevoir une information du clavier, le 6804 descend le bit PC3 à 0, ce qui provoque une interruption par le bit CP1 du 6846. Il attend ensuite que le 6809 descende le fil P5 du 6846 à 0, lui indiquant qu'il est prêt à recevoir les 9 bits de données (signal ACK).

Les informations séries transmises répondent à la convention suivante:

– un 1 logique est codé comme étant une impulsion positive de 56 microsecondes.
– un 0 logique est codé comme étant une impulsion positive de 38 microsecondes.

Dans l'exemple de la figure 25, la touche CNT est enfoncée en même temps que la touche "U" du clavier "métropole" dont le code est 32 en hexadécimal.

• Emission d'une requête au clavier par le 6809 E:

Le 6809 peut émettre 3 requêtes différentes au 6804:

Initialisation: Le 6804 renvoie alors au 6809 E, un code indiquant que le clavier est configuré en version Française ou Export. Le clavier se met en CAPS LOCK actif - LED allumée.

Majuscule: Le clavier se met en CAPS LOCK actif - LED allumée.

Minuscule: Le clavier se met alors en CAPS LOCK inactif - LED éteinte.

Quand le 6809 E souhaite émettre une requête, il met P5 à 0 et attend que le 6804 descende de fil PC3 à 0, le 6804 compte alors le temps pendant lequel P5 reste à 0. Le 6809 E peut donc générer trois temporisations différentes correspondant aux trois requêtes possibles:

– 0,67 milliseconde --> demande d'initialisation du 6809 E
– 1,30 milliseconde --> mise en CAPS LOCK actif
– 1,90 milliseconde --> mise en CAPS LOCK inactif.

9. Le contrôleur de l'unité de disquette

Le contrôleur de drive du TO8 est un nouveau boîtier gate array THMFC1 développé par THOMSON Micro-informatique, pour répondre aux besoins des nouvelles machines. Il présente, notamment, une adaptabilité aux différents formats d'enregistrement des données (codage FM et MFM) et à la gestion des floppies 3"5, 5"25 ou d'un QDD (Quick Disk Drive).

Il peut commander deux unités de disquettes à la fois. D'un point de vue externe, le THMFC1 s'apparente au WD 2793 ou WD 1770 utilisés sur le TO9. En interne, il se distingue par l'emploi d'un séparateur de données utilisant la technique de comptage à la place d'un circuit à verrouillage de phase (PLL). De même, il intègre un registre du choix de codage simple ou double densité (FM, MFM).

Branchements du THMFC1

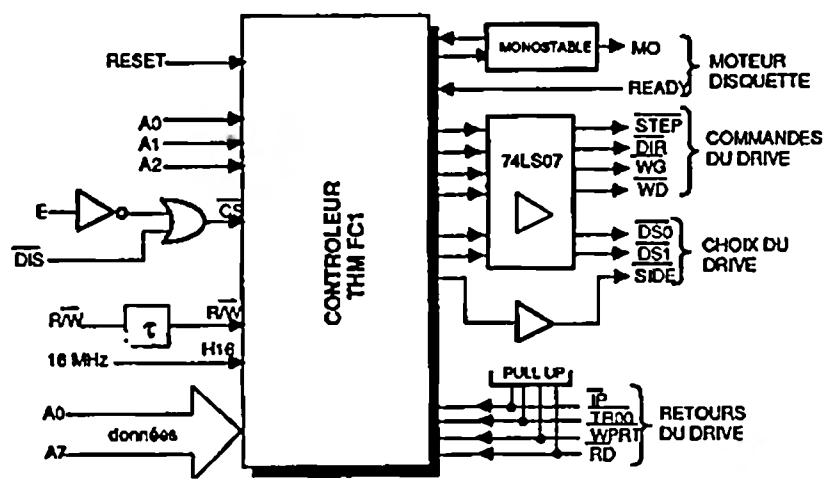


Figure 26. Le gate array contrôleur de disques

En liaison directe avec le bus du 6809 E (figure 26), le contrôleur est connecté au bus d'adresses par les trois lignes A2, A1, A0. Ces trois bits, en relation avec le décodage d'adresse DISN (actif pour E = 1) dans l'espace E7D0 à E7D9, et en relation avec la commande R/WN (retardée en lecture), permettent la sélection des onze registres dont dispose le composant.

La figure 26 décrit les différentes commandes concernant les unités de disquettes.

On distingue:

– La commande MO du moteur de drive (non utilisée pour le QDD). Elle est en liaison avec la sélection du contrôleur qui, via un monostable redéclenchable, avec une constante de temps de plusieurs secondes, intègre les demandes d'accès et procure une inertie suffisante du moteur adaptée pour le bon fonctionnement du mécanisme.

– READY est une information en retour de la mécanique confirmant l'insertion de la disquette, la mise en route du moteur, la fréquence d'index correcte. Cette commande permet de valider, dans le contrôleur, les actions de lecture et d'écriture.

Des commandes bufferisées par un 74 LS 07 délivrent:

– Les signaux de commande mécanique du drive:

STEPN — Pour déplacer la tête de lecture/écriture d'un pas. Cette commande est inutilisée dans un QDD.

DIRN — Pour indiquer à la mécanique dans quel sens déplacer la tête (0 vers le centre, 1 vers l'extérieur).

WGN — Le signal d'autorisation d'écriture.

WDN — Flot de données séries d'écriture de la disquette.

DS0N DS1N — Les signaux de choix du drive.

Le signal SIDEN bufferisé par un transistor indique à la mécanique sur quelle face de la disquette travailler (0 face supérieure, 1 face inférieure).
Pour le QDD, cette commande concerne l'alimentation du moteur.

Quatre signaux d'entrée informent le contrôleur:

IPN — Indique le passage du repère d'index (une impulsion négative par tour de disquette). Dans le QDD, ce bit indique la présence ou l'abscence de disquette (1 = disquette présente).

TROON — Indique que la tête de lecture écriture est positionnée sur la piste 0. Avec le QDD, ce bit est envoyé constamment à l'état 0, ce qui permet au THMFC1 de "savoir" quel type de machine il doit contrôler.

WPRTN — Indique que la disquette insérée dans la mécanique est protégée en écriture.

RDN — Flot de données lues sur la disquette.

Description et programmation des registres

• Registre CMD0 en écriture à E7D0

bit	7 à 0		
bit	6 à 0		
bit	5 à 0	—>	MFM
	à 1	—>	FM
bit	4 à 1	—>	validation de la détection des mots de synchro
bit	3 à 0	—>	inhibition de la synchronisation du séparateur de données pour le formatage.
bit	2 à 1	—>	active la sortie WGN
bit	1		
bit	0	—>	code opération

0	0	reset
0	1	écriture secteur
1	0	lecture adresse
1	1	lecture secteur

• Registre CMD1 en écriture à E7D1

bit	7	—>	bit de compatibilité
bit	6		
bit	5	—>	longueur du secteur

0	0	128 mots/secteur
0	1	256 mots/secteur
1	0	512 mots/secteur
1	1	1024 mots/secteur

bit	4 à 0	—>	face 0 du disque
	à 1	—>	face 1
bits	3, 2, 1	—>	commande de précompensation à 437,5 ns par pas de 62,5ns.
bit	0 à 1	—>	inhibition du système, lorsque le signal READY est inactif (bit à 1).

• Registre CMD2 en écriture à E7D2

Registre de commande, il a des fonctions différentes selon le drive utilisé:

Floppy ou QDD.

bit	7	—>	non utilisé.
bit	6 à 0	—>	face 0 du floppy.
	0	—>	commande active du moteur QDD.
	1	—>	fonctions inverses.
bit	5 à 0	—>	commande de direction vers l'extérieur (piste 0) du disque.
	1	—>	fonction inverse non utilisé pour le QDD.
bit	4 à 0	—>	commande de pas inactivée pour le floppy.
	1	—>	active. non utilisé pour le QDD.
bit	3	—>	non utilisé
bit	2 à 0	—>	commande moteur inactive pour le floppy.
	1	—>	active. non utilisé pour le QDD.
bit	1	—>	commandes de sélection de drive (floppy
bit	0		et QDD) actives à l'état 1.

• Registre d'état STAT0 en lecture à E7D0.

bit	7	—>	image de l'horloge caractère:
	à 1	—>	demande d'opération
	à 0	—>	par lecture ou écriture des registres RDATA ou WDATA.
bit	6 à 0		
bit	5 à 0		
bit	4 à 1	—>	indication que l'opération se termine.
bit	3 à 1	—>	indication que l'opération est terminée.
bit	2 à 1	—>	erreur de CRC (check sum de la zone d'identification de données).
bit	1	—>	action identique au bit 7 pour les opérations dites intelligentes.
bit	0 à 1	—>	indique le bon résultat d'une détection synchro.

• Registre d'état STAT1 en lecture à E7D1

Ce registre contient des informations différentes selon le drive utilisé.

bit	7 à 0		
bit	6 à 1	—>	détection d'index pour le floppy.
	à 1	—>	présence de disquette pour de QDD.
bit	5 à 1	—>	information de changement de disquette non utilisé pour le QDD.
bit	4	—>	image inverse de la commande moteur MO.
bit	3 à 1	—>	détection de la piste 0 pour le floppy.
	à 1	—>	information de détection d'un QDD.
bit	2 à 1	—>	information de protection en écriture sur le floppy et le QDD.
bit	1 à 1	—>	information "ready" en provenance du floppy ou QDD.
bit	0	—>	non utilisé.

• Registres de données WDATA, RDATA en écriture ou lecture à E7D3.

Ces registres 8 bits ont le rôle traditionnel de tampon.

• Registre d'horloge type en écriture à E7D4.

Ce registre 8 bits contient la configuration (FF) pour les données et (0A) pour les mots de synchro.

• Registre secteur WSECT en écriture à E7D5.

Comporte le numéro de secteur à chercher. Le contrôleur se charge de comparer les informations écrites dans ce registre avec celles présentes sur la disquette dans la zone d'identification.

• Registre piste WTRCK en écriture à E7D6.

Même gestion que pour le registre secteur.

• Registre largeur de cellule WCELL en écriture à E7D7.

bit	7 à 0	—>	modification des caractéristiques du séparateur de données (pour pistes intérieures de la disquette. Précompensation).
	à 1	—>	fonctionnement normal du séparateur.
bits	6-0	—>	valeur permettant de charger le compteur du séparateur selon le mode de codage utilisé.

Spécification d'un secteur

Un secteur est composé d'un champ d'identification et d'un champ de données selon le modèle:

	Nombre d'octets	Caractères	Désignation
Champ d'identification	12	0 0	synchro bit
	3	A1 horloge 0A	synchro caractères
	1	F E	adresse début identif.
	1		numéro de piste
	1		numéro de face
	1		numéro de secteur
	1		longueur secteur
	2		contrôle CRC
	22		espaces
Champ de données	12	0 0	synchro bit
	3	A1 horloge 0A	synchro caractère
	1	F B	adresse début donnée
	selon longueur de secteur		données
	2		contrôle CRC
	1		inhibition porte écrit.
	variable selon le drive		espaces

Les fonctions "intelligentes" du contrôleur consistent à lire une adresse, un secteur ou écrire un secteur. Elles sont programmées par les bits 1 et 0 du registre CMD0. Chaque fonction doit permettre l'acquittement d'une série d'opérations répondant aux spécifications décrites ci-avant.

Quatrième partie

Analyse matérielle du TO9+

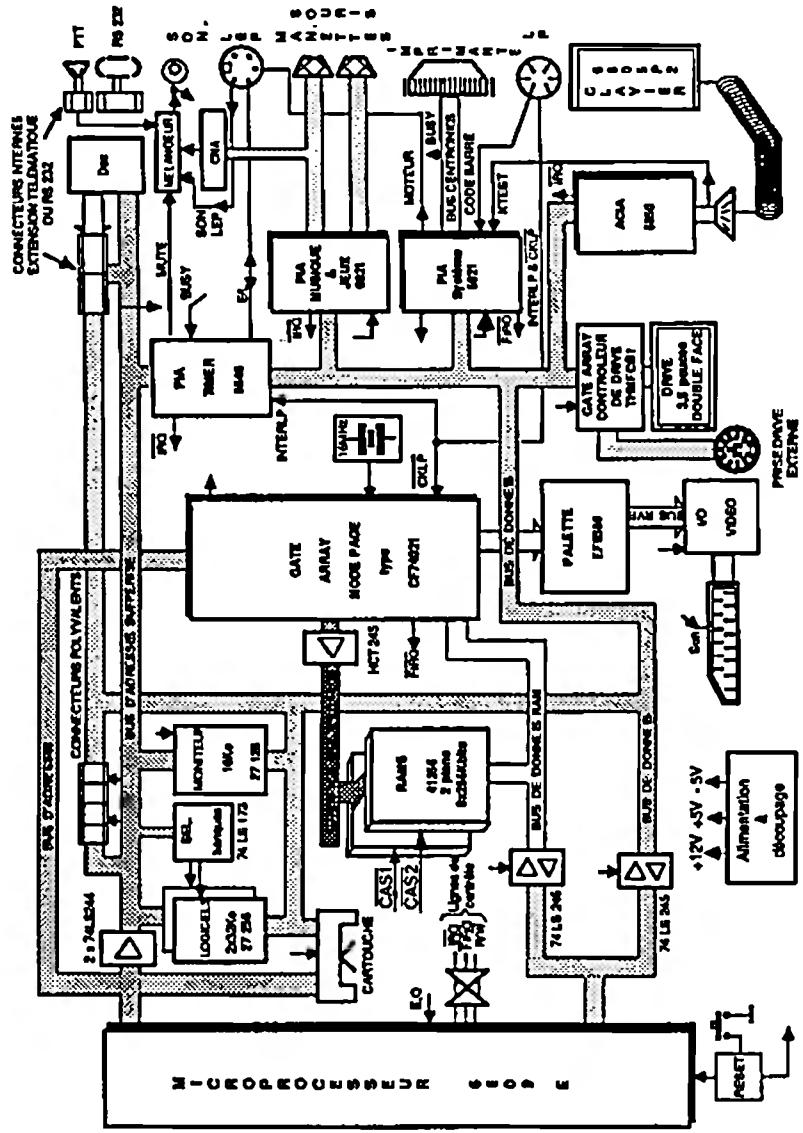


Figure 27. Synoptique de l'unité centrale TO9+

1. Conception générale Description

Le TO9+ étant la symbiose du TO9 et du TO8, nous faisons référence tout au long de cette partie à des similitudes de structure et de fonctionnement. Nous invitons le lecteur à se reporter, pour plus de détails, aux chapitres concernant les produits cités en comparaison.

Comme le TO9, le micro-ordinateur TO9+ est architecturé autour du 6809 E, microprocesseur commandé à la fréquence de 1 MHz par deux horloges extérieures en quadrature: E et Q.

Le bus d'adresses 16 bits est direct pour la cartouche et le gate array mode page. Il est bufferisé par 2 × 74 LS 244 pour adjoindre les autres composants.

Le bus de données 8 bits comporte deux dérivations bufferisées et contrôlées par deux 74 LS 245. Les informations sont alors aiguillées, selon leur destination, sur le bus de données RAMS ou sur le bus de données afférent aux principaux composants de l'unité. En dehors de l'aiguillage RAM, ce principe permet de "délester" le 6809, compte tenu des nombreux circuits à alimenter.

Le bus de données RAM assure, plus particulièrement en mode page, les transferts d'information entre les RAMS et le gate array.

Comme pour le TO9,les lignes de contrôles du 6809 E correspondent:
- aux commandes de lecture écriture R/WN des différents registres et mémoires,
- aux demandes d'interruption IRQN concernant la gestion du clavier, le clignotement du curseur, les manettes de jeux et la souris,
- aux demandes d'interruption FIRQN pour le fonctionnement du crayon optique et du code barre.

Un circuit de réinitialisation "RESET" est en relation avec le 6809 E et les PIA.

D'une manière identique au TO8, la mémoire morte comporte:

- Les deux pages de 6 Ko du moniteur ainsi que les deux pages de 1,9 Ko de logiciel contrôleur de disque qui sont logées dans une EPROM 16 Ko 27 128. Chacune des pages est sélectée à partir du bit P4 du PIA 6846.

- Les 2 × 32 Ko de logiciel d'application BASIC 1 - divers - BASIC 512 - EXTRAMONITEUR, qui sont logés chacun dans une ROM ou EPROM 32 Ko selon une répartition en quatre banques. D'une manière semblable au TO8, la commutation des banques s'effectue en programmation par une écriture d'adresses dans un LATCH 74 LS 173. Ce circuit permet de sélecter chaque partie concernée dans une des mémoires mortes.

La cartouche de logiciel d'application externe est sélectée par logiciel à partir du bit P2 du PIA 6846.

La mémoire vive est composée de deux plans de 256 Ko chacun:

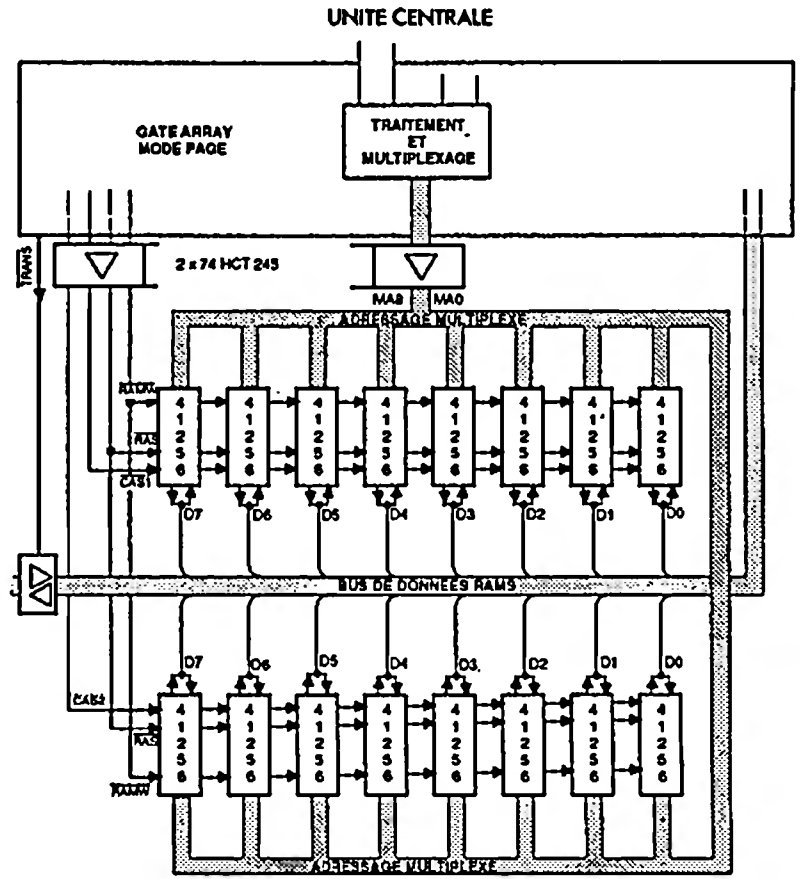


Figure 28. Système de mémorisation RAM dans le TO9+

Huit boîtiers 256 K-bits 41256 forment un premier plan mémoire validé par CAS1N et comprenant les différentes pages de 16 Ko initialement telles que:

page 0	=	mémoire ou espace "écran" avec la RAMA et la RAMB
page 1	=	mémoire ou espace "système" avec la page moniteur
pages 2 à 15	=	banques mémoires utilisateur ou espace "données".

Les différentes pages sont commutées par un adressage physique sur 18 bits (MA8-MA0), en provenance du gate array mode page.

Le deuxième plan mémoire est validé par CAS2N. Il correspond à l'extension mémoire dans le TO8 et regroupe, par ses huit boîtiers 41256, seize pages de RAM utilisateur (pages 16 à 31). Adressé de la même manière que le premier plan, il représente la suite de l'espace "données" (cf. figure 28).

Conformément au TO8, le TO9+ utilise le gate array mode page CF 74021 ou EFG 202A pour lequel il joue un rôle parfaitement identique. Il est en relation avec les plans RAM dynamiques par l'intermédiaire du bus d'adresses multiplexées. Ce dernier est adapté par un HCT245. Les commandes des boîtiers mémoires sont elles-mêmes adaptées par un HCT245.

La sélection du bus de données RAM est dépendante du signal TRANSN par l'intermédiaire du 74 LS 245. Il assure l'aiguillage des informations et permet notamment un accès du CPU dans l'espace mémoire: 0000-3FFF (cf. Le 6809 dans le TO8, page 95). Un signal complémentaire de TRANSN agit de façon opposée sur le 74 LS 245 du bus de données principal.

Le gate array mode page détermine les mêmes cas de fonctionnement ou figures réalisables sur TO8 dont, plus particulièrement, les recouvrements d'espace mémoires. De par le registre E7DC, il procure, en relation avec la RAMA et la RAMB, les mêmes modes d'affichage connus sur le TO8 ou le TO9 (cf. Le système de visualisation du TO9, page 52).

Les signaux, en provenance du crayon optique, subissent le même traitement que sur le TO8.

Toujours conformément au TO8, un circuit de palette du type EF 9369, programmable par le 6809 E, permet le choix des seize teintes exploitables parmi 4 096. Il distribue les informations nécessaires à la prise péritel, via des circuits d'interfaçage vidéo.

Deux connecteurs de carte "polyvalents" établissent les liaisons bus et signaux nécessaires aux extensions.

Quatre circuits d'interface sont en relation avec différents périphériques:

– Un PIA TIMER 6846 qui, exception faite de la ligne de contrôle CP1 et du bit P5 devenus inutilisés, est câblé et fonctionne comme sur le TO8.

– Un PIA 6821 "musique et jeux" pour l'élaboration du son et les commandes de manettes et souris; un PIA "système". De par leur fontionnement, ils sont parfaitement identiques aux circuits utilisés sur le TO8. Ainsi, le bit de forme et les bits de commutation de banques mémoires du TO9 voient leur action totalement émulée par le gate array mode page dans le TO9+.

– Un ACIA 6850 qui, comme sur le TO9, réalise la transformation série/parallèle des informations envoyées en série à travers le cordon du clavier.

Il existe cependant une différence notable qui singularise, en ce point particulier, le fonctionnement du TO9+: la transmission série asynchrone dans le TO9 devient synchrone pour le TO9+. Pour ce faire, l'horloge est fournie avec les données série par un fil supplémentaire en provenance du monochip du clavier. La liaison entre le clavier et la partie unité centrale est bi-directionnelle. La cadence de travail est d'environ 9600 bauds.

Les registres de l'ACIA sont accessibles aux adresses suivantes:

E7DE	registre de contrôle en écriture
E7DE	registre d'état en lecture
E7DF	registre d'emission en écriture
E7DF	registre de réception en lecture.

La figure 29, page suivante, traduit le fonctionnement matériel du clavier.

L'élément de dialogue est un monochip 6805 P2 dont le port A et le port B réalisent l'exploration de la matrice clavier. Le port C est en partie consacré aux échanges avec l'ACIA. Ce microprocesseur est piloté par une horloge de 4 MHz. Une commutation interne offre la possibilité de monter un 68705.

Le programme du 6805 P2 est tel que, à chaque appui sur une touche, un octet précédé d'un bit de start et suivi par un bit de stop, est envoyé après un délai maximum de 8 ms (temps de scrutation). La prise en compte de chaque bit se fait sur un front montant de l'horloge RXC fournie par le 6805 P2 (transmission synchrone).

Si cette touche reste enfoncée, après un délai maximum de 0,8 s, le code est renvoyé toutes les 70 ms (répétition automatique de 14 caractères par seconde). Un signal KTEST est expédié vers l'unité centrale pour indiquer l'action effective de la touche enfoncée. Ce signal est actif environ 100 ns après l'appui d'une touche, alors que le temps de cycle de scrutation du clavier est de 8 ms.

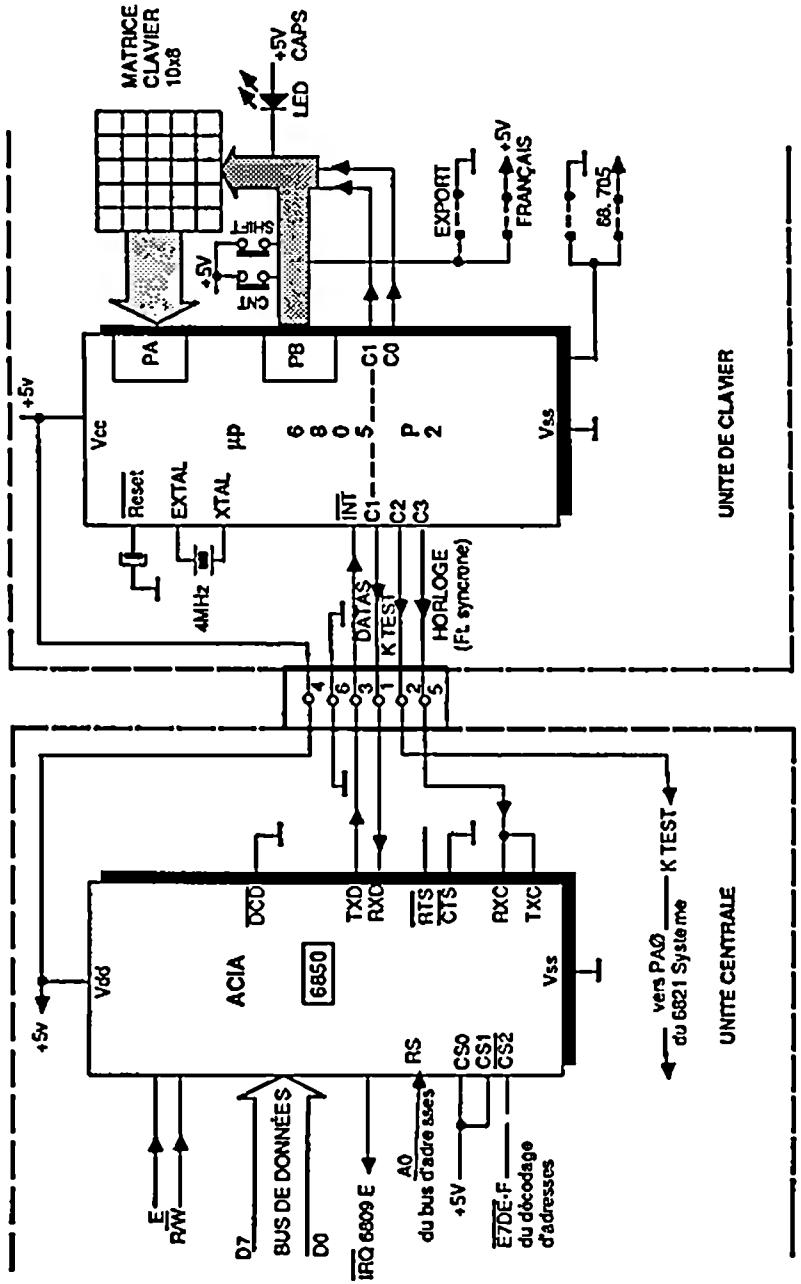


Figure 29. Structure matérielle du fonctionnement du clavier dans le TO9+

Comme pour le TO9, le clavier est susceptible de recevoir des informations en provenance de l'unité centrale. La transmission est composée de trois bits avec un bit de start et un bit de stop. Le bit de start est envoyé lorsque l'ACIA reçoit une impulsion d'horloge TXC. Lorsque le bit de start est présent, une interruption externe est déclenchée et le mot est lu par le monochip.

Le gate array contrôleur de disquettes THMIC01 équipant le TO8 est également utilisé sur le TO9+. Il gère l'unité de floppy 3,5 pouces intégrée et délivre, en parallèle, par l'intermédiaire d'une prises DIN 14 broches, les signaux nécessaires au fonctionnement d'une drive externe selon les deux standards 5,25 pouces ou 3,5 pouces.

Une alimentation à découpage semblable à celle du TO9 fabrique les trois tensions de + 5 V, de + 12 V et - 5 V pour le fonctionnement des composants de l'unité centrale et des extensions.

2. Extension intégrée

Le TO9+ est équipé, en relation avec le décodage d'adresse correspondant, d'un module extension télématique qui permet de relier directement l'ordinateur au réseau téléphonique connecté selon la norme NFC 98010 et les spécifications techniques du CNET 1108 et 1435. Le principe de transmission est conforme à l'avis V23 du CCITT.

• Conception de l'extension télématique

Elle est accessible aux adresses E7F8 à E7FF. Elle est composée:

– D'un ACIA 6850 qui réalise la sérialisation des données en émission et leur désérialisation en réception.

– D'un 6821 qui assure les commandes et détections nécessaires au modem et à l'interface ligne (prise de ligne, numérotation, détection d'appel, retournement et passage du modem en half duplex...).

– D'un MODEM EFB 7513: modulateur/démodulateur FSK V23, pouvant assurer un dialogue "full duplex" car les voies émission et réception sont transmises par des porteuses de fréquences différentes.

– D'un duplexeur constitué d'un transformateur et amplificateurs opérationnels TDB 0124, pour mixer et séparer les signaux émis et reçus par la ligne téléphonique bi-directionnelle. Cette fonction est appelée conversion 2 fils/4 fils.

– D'une interface ligne (relais optocoupleurs et transformateur) qui permet d'adapter les signaux aux exigences de transmission et de sécurité du réseau téléphonique commuté des PTT (isolement galvanique par rapport au secteur, régulation du courant de ligne, numérotation décimale et détection d'appel).

Elle peut être gérée par un logiciel appelé handler télématique et qui respecte les normes en vigueur.

Cinquième partie

Le moniteur

1. Généralités

Les moniteurs des TO8, TO9 et TO9+ renferment un ensemble de programmes appelés routines, écrits en langage machine. Le rôle de chacune d'elles est de gérer une fonction de la machine. Ainsi nous trouverons une routine chargée de visualiser des caractères sur l'écran, une autre fera la scrutation du clavier pour éventuellement détecter une touche appuyée, une troisième permettera d'utiliser les manettes de jeux, etc.

Les programmes integrés par exemple dans le TO9, tels le BASIC ou PARAGRAPHE, utilisent ces routines pour effectuer leurs différents traitements; un utilisateur muni de la cartouche ASSEMBLEUR peut en faire autant.

La démarche, très générale, de l'utilisation d'une routine s'opère en trois temps:

1) Vous chargez avec des valeurs ayant une signification précise certains registres du 6809 E, et éventuellement certains registres en RAM. Cette opération s'appelle l'initialisation des paramètres d'entrées.

2) Vous appelez la routine du moniteur qui effectue l'opération desirée.

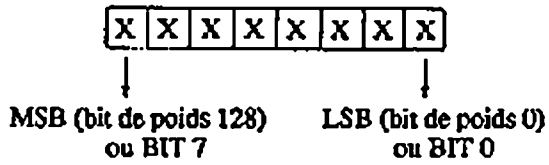
3) Vous recupérez, s'il y a lieu, le résultat de l'opération dans certains registres du 6809 E, et/ou dans des registres en RAM. Ce sont les paramètres de retour.

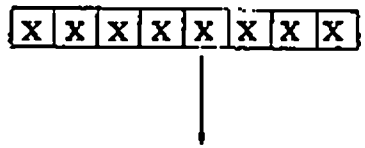
Vous verrez que, dans certains cas, il n'y a pas de paramètres de retour. Quand ils existent, ils peuvent se trouver soit dans des registres du microprocesseur, soit dans des registres en RAM, soit dans les deux.

Dans le cadre de cette étude logicielle, des valeurs sont exprimées en décimal ou en hexadécimal. Pour faire la différence, nous avons employé la même convention d'écriture que la cartouche ASSEMBLEUR: si aucun signe ne précède le chiffre, la valeur est en décimal. Par contre, si le chiffre est précédé du symbole "$", le nombre est écrit en hexadécimal.

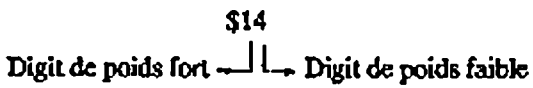
Exemple: 12 = $0C

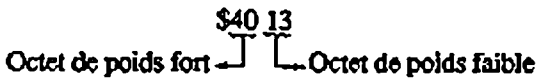
La majeure partie des codes ou valeurs employés sont exprimés sur un octet (8 bits) ou un double octet (16 bits). Nous utiliserons en conséquence les termes suivants:





Les appellations "4e LSB" ou "BIT 3" designent toutes les deux le bit repéré ci-dessus.





Certains registres en RAM recevront des valeurs cadrées sur un octet, d'autres sur deux octets. Dans le second cas, une consigne indiquant qu'il faut par exemple mettre la valeur $13F dans le registre TOTO ($6030-$6031) signifie que:

1) La valeur $01 doit être implantée à l'adresse $6030
2) La valeur $3F doit être implantée à l'adresse $6031.

L'accès normalisé à une routine du moniteur se fait en utilisant l'instruction "JSR" ou "JMP", suivie de l'adresse du point d'entrée (ces adresses sont au nombre de 24). Souvent des exemples sont donnés sous cette forme:

```
        .
        .
PUTC    EQU     $E803
        LDB     #$07
        JSR     PUTC
        .
        .
```

Les points signifient que des instructions peuvent être avant ou après, il ne s'agit alors que d'une bribe de programme. Si vous voulez "essayer" l'exemple, il convient de rajouter simplement un arrêt (par exemple SWI), et les directives d'assemblage ORG et END.

Exemple: reprise et complément du programme précédent:

```
        ORG     $A000
PUTC    EQU     $E803
        LDB     #$07
        JSR     PUTC
        SWI
        END
```

Les sous-programmes du moniteur sauvegardent les registres du 6809 E. Au retour, tous les registres sont remis dans l'état antérieur de l'appel, sauf le registre code condition du 6809 E (ou registre d'état RE) et, bien sûr, les registres contenant des paramètres de retour (le plus souvent B, X et Y).

Il vous est **fortement conseillé** d'appeler les routines du moniteur en utilisant l'accès normalisé. En effet, ces routines ne peuvent fonctionner correctement qu'avec le pointeur de pile et certains registres du 6809 E positionnés dans un état bien précis.

2. Gestion alphanumérique de l'écran

Générateur de caractères alphanumériques

Les TO8, TO9 et TO9+ possèdent trois générateurs de caractères conciliant souplesse, rapidité et facilité d'utilisation. Le point commun de ces générateurs est, bien sûr, la manière d'afficher un caractère dans la fenêtre de travail. La méthode est la suivante: chaque caractère est défini par une matrice de 8 × 8 points, soit 64 points mémorisés par 8 octets de mémoire consécutifs. La codification d'un octet répond à la norme ci-dessous:

0 = le point n'est pas allumé
1 = le point est allumé.

Par exemple, le tracé du caractère A se définira par la matrice suivante:

```
MSB      LSB
 ↓        ↓
0 0 0 0 0 0 0 0  8eme octet
0 0 0 1 1 0 0 0  7eme octet
0 0 1 0 0 1 0 0  6eme octet
0 1 0 0 0 0 1 0  5eme octet
0 1 1 1 1 1 1 0  4eme octet
0 1 0 0 0 0 1 0  3eme octet
0 1 0 0 0 0 1 0  2eme octet
0 0 0 0 0 0 0 0  1er octet
```

Sa codification dans le générateur de caractères étant dans l'ordre les octets $00, $42, $42, $7E, $42, $24, $18, $00.

Alphabet standard G0

Le générateur de caractères G0 est celui implicitement utilisé par les TO8, TO9 et TO9+. Il est constitué d'une suite de caractères affichables, correspondants au standard ASCII. L'adresse de ce générateur est contenue en RAM dans le registre appelé PTGENE et situé en $60CF-$60D0.

De fait, vous avez la possibilité de créer votre propre générateur de caractères selon le principe énoncé ci-dessus, et de l'implanter en RAM. Vous demanderez ensuite au microprocesseur de travailler avec votre police de caractères, en implantant l'adresse de ce nouveau générateur dans le registre PTGENE. Les seules contraintes à respecter sont de débuter ce générateur par le caractère correspondant au code $20, le second sera appelé par le code $21, etc. D'autre part, ne pas oublier qu'un caractère est constitué de 8 octets consécutifs, et enfin que le premier octet correspond au bas du caractère, le 8eme au haut du caractère.

Alphabet G2

Cet alphabet suit et complète le générateur G0. Il est composé de 22 éléments permettant l'affichage des caractères accentués (aigus, graves et circonflexes), ainsi que des configurations spéciales comme le tréma et la cédille.

Caractères utilisateurs

Indépendamment des alphabets G0 et G2 pouvant être redéfinis, les TO8, TO9 et TO9+ nous offrent la possibilité de travailler avec des caractères dits "utilisateurs" qui, seuls ou associés, peuvent aussi bien permettre d'afficher du texte que des dessins.

Ce générateur est en BASIC initialisé par la fonction DEFGR$ et sollicité par PRINTGR$. Le principe de création et de mémorisation d'un caractère est identique à celui décrit précédemment. L'adresse du début de ce générateur de caractères utilisateur doit être implanté dans le registre USERAF ($602D-$602E). Le premier caractère (constitué des 8 premiers octets) correspondra automatiquement au code $80, le second au code $81 et ainsi de suite jusqu'à $FF. La taille maximum de ce générateur est donc de 128 caractères.

Affichage des caractères alphanumériques

La routine PUTC permet d'afficher, à la position courante du curseur d'écran, un caractère contenu dans l'un des différents générateurs de caractères décrits précédemment. Le code du caractère desiré devra être implanté dans l'accumulateur B du CPU juste avant l'appel à la routine. Les codes compris entre $20 et $7F appelleront un caractère inclus dans l'alphabet pointé par PTGENE ($60CF-$60D0). Théoriquement, il s'agit donc des générateurs G0 et G2, mais nous avons vu qu'il était également possible que ces codes appellent un caractère d'une police redéfinie.

Les codes compris entre $80 et $FF correspondent aux caractères "utilisateur". Le générateur sollicité sera celui dont l'adresse de début est implantée dans le registre USERAF ($602D-$602E).

Le tableau ci-dessous dresse la liste des codes hexadécimaux et des caractères correspondants pour les alphabets G0 et G2.

Code	Caract.	Code	Caract.	Code	Caract.
20	blanc	40	@	60	—
21	!	41	A	61	a
22	"	42	B	62	b
23	#	43	C	63	c
24	$	44	D	64	d
25	%	45	E	65	e
26	&	46	F	66	f
27	'	47	G	67	g
28	(	48	H	68	h
29	)	49	I	69	i
2A	*	4A	J	6A	j
2B	+	4B	K	6B	k
2C	,	4C	L	6C	l
2D	-	4D	M	6D	m
2E	.	4E	N	6E	n
2F	/	4F	O	6F	o
30	0	50	P	70	p
31	1	51	Q	71	q
32	2	52	R	72	r
33	3	53	S	73	s
34	4	54	T	74	t
35	5	55	U	75	u
36	6	56	V	76	v
37	7	57	W	77	w
38	8	58	X	78	x
39	9	59	Y	79	y
3A	:	5A	Z	7A	z
3B	;	5B	[	7B	{
3C	<	5C	\	7C	|
3D	=	5D	]	7D	}
3E	>	5E	^	7E	~
3F	?	5F	-	7F	■

Nom	: PUTC
Adresse du point d'entrée	: $E803
Paramètre d'entrée	: Accumulateur B du 6809E
Paramètre de retour	: Néant
Effet	: Cette routine affiche à l'écran le caractère correspondant au code envoyé. Les valeurs sont comprises entre $20 et $7F.

Exemple:

```
* AFFICHAGE DU GENERATEUR DE CARACTERE
* GO

        TITLE   GENCARAC
        ORG     $A000
PUTC    EQU     $E803
        LDB     #$20        20=1 CARACT
SUIT    JSR     PUTC        AFFICH CARACT
        INCB
        CMPB    #$80        7F=DERNIER CARACT
        BNE     SUIT
        SWI
        END
```

Positionnement des caractères

Hormis les valeurs comprises entre $20 et $FF correspondant à des codes de caractères alphanumériques affichables à l'écran, la routine PUTC interprète également les valeurs $07 à $1F. Elles déterminent des modes de traitement particulier de PUTC: création d'un bip sonore, séquence d'échappement... En particulier, le code US ($1F) déclenche une séquence de positionnement du curseur ou de définition de la fenêtre de travail.

Programmation du curseur

Les coordonnées du curseur sont déterminées par la ligne (L) et la colonne (C). Alors que L est toujours compris entre 0 et 24, l'intervalle de C est fonction du mode de visualisation:

En mode 40 colonnes $1 \leq C \leq 40$ décimal
En mode 80 colonnes $1 < C < 80$ décimal

Une séquence de positionnement du curseur s'opère en trois appels de PUTC:

1e appel consiste à envoyer le code US ($1F)
2e appel envoie le numéro de ligne
3e appel envoie le numéro de colonne.

Les numéros de ligne et de colonne envoyés à PUTC se calculent selon les formules suivantes:

ligne = L + $40 L est exprimé en base 16
colonne = C + $40 C est exprimé en base 16.

Exemple: On désire positionner le curseur sur la ligne 12 et la colonne 17:
12 en décimal correspond à $0C en hexadécimal
17 en décimal correspond à $11 en hexadécimal

ligne = $0C + $40 = $4C
colonne = $11 + $40 = $51

La séquence s'écrira donc:

```
        .
        .
PUTC    EQU   $E803
        LDB   #$1F     code US envoyé
        JSR   PUTC     1e appel
        LDB   #$4C     numéro ligne
        JSR   PUTC     2e appel
        LDB   #$51     numéro colonne
        JSR   PUTC     3e appel
        .
        .
```

Nom	: PUTC
Adresse du point d'entrée	: $E803
Paramètre d'entrée	: Accumulateur B du 6809 E
Paramètre de retour	: Néant
Effet	: Pour des valeurs comprises entre $07 et $1F les effets sont les suivants:

Code hexa	Nom	Effet
$07	BEL	Création d'un bip sonore
$08	BS	Déplacement curseur d'une position à gauche, ou recopie à gauche du caractère courant si l'adresse $6043 contient la valeur $FF
$09	HT	Déplacement curseur d'une position à droite, ou recopie à droite du caractère courant si l'adresse $6043 contient la valeur $FF
$0A	LF	Descente d'une ligne
$0B	VT	Remontée d'une ligne
$0C	FF	Effacement de la fenêtre
$0D	CR	Retour en début de ligne courante
$0E	SO	Passage en mode vidéotel
$0F	SI	Retour en mode normal
$13	DC1	Allumage du curseur
$12	DC2	Répétition du dernier caractère ASCII affiché
$14	DC4	Extinction du curseur
$16	ACC	Séquence caractère du G2
$18	CAN	Effacement d'une ligne à partir de la position du curseur
$1B	ESC	Séquence d'échappement
$1E	RS	Retour du curseur dans le coin supérieur gauche de la fenêtre
$1F	US	Séquence de positionnement curseur ou de définition de fenêtre

Détermination de la fenêtre de travail

La fenêtre de travail sera définie par le repérage de la première ligne ou "haut de page" et de la dernière ligne ou "bas de page". L'un comme l'autre nécessitent trois appels à la routine PUTC:

1e appel: envoi du code US ($1F) Accu B
2e appel: envoi du nombre N1 par l'accu B
3e appel: envoi du nombre N2 par l'accu B.

Haut de page

La ligne représentant le haut de page est repérée par un nombre compris entre 00 et 24 (décimal), donc s'écrivant par deux digits. La programmation du haut de page se fera en décomposant ces deux digits pour former N1 et N2 selon la formule suivante:

N1 = digit de poids fort + $20
N2 = digit de poids faible + $20

Ainsi N1 et N2 seront toujours compris entre $20 et $29. Prenons un exemple: On désire que le haut de la fenêtre soit la ligne 01:

N1 = 0 + $20 = $20
N2 = 1 + $20 = $21

La séquence s'écrira:

```
        .
        .
PUTC    EQU     $E803
        LDB     #$1F    code US
        JSR     PUTC    1e appel
        LDB     #$20    N1
        JSR     PUTC    2e appel
        LDB     #$21    N2
        JSR     PUTC    3e appel
          .
          .
```

Bas de page

La ligne représentant le bas de page est un nombre décimal compris entre 00 et 24, donc constitué de deux digits. La programmation du bas de page se fera en décomposant ces deux digits pour former N1 et N2 selon la formule ci-dessous:

N1 = digit de poids fort + $10
N2 = digit de poids faible + $10

Ainsi N1 et N2 seront compris entre $10 et $19.

Exemple: on désire que le bas de la fenêtre soit la ligne 24:

le digit de poids fort est 2
le digit de poids faible est 4.

Donc,
N1 = 2 + $10 = $12
N2 = 4 + $10 = $14

La séquence s'écrira:

```
        .
        .
PUTC    EQU     $E803
        LDB     #$1F
        JSR     PUTC
        LDB     #$12
        JSR     PUTC
        LDB     #$14
        JSR     PUTC
          .
          .
```


Retour du curseur dans le coin gauche

Pour positionner le curseur dans le coin supérieur gauche de la zone définie comme étant la fenêtre de travail, vous utiliserez le code $1E, envoyé à PUTC de la manière suivante :

```
        .
        .
PUTC    EQU     $E803
        LDB     #$1E
        JSR     PUTC
          .
          .
```

Descente d'une ligne

Il suffit d'envoyer le code $0A à la routine PUTC pour provoquer la descente du curseur d'une ligne. La colonne reste inchangée.

```
        .
        .
PUTC    EQU     $E803
        LDB     #$0A
        JSR     PUTC
          .
          .
```

Remontée d'une ligne

L'opération inverse à celle décrite plus haut est réalisée en envoyant le code $0B à la routine PUTC.

```
        .
        .
PUTC    EQU     $E803
        LDB     #$0B
        JSR     PUTC
          .
          .
```

Retour au début de ligne

Le code $0D implanté dans l'accu B provoque, lors de l'appel de PUTC, un positionnement du curseur sur la colonne 0 de la ligne courante.

Effacements divers

Effacement de la fenêtre

Cette opération est réalisée en envoyant le code $0C à la routine PUTC.

```
        .
        .
PUTC    EQU     $E803
        LDB     #$0C
        JSR     PUTC
        .
        .
```

Tout ce qui est dans la zone définie comme étant la fenêtre de travail sera alors entièrement effacé.

Extinction et allumage du curseur

Le code $14 envoyé à la routine PUTC aura pour effet d'effacer le curseur de l'écran.

```
        .
        .
PUTC    EQU     $E803
        LDB     #$14
        JSR     PUTC
        .
        .
```

Pour rallumer ce curseur, vous utiliserez le code $11.

Effacement d'une ligne

Le code $18 envoyé à la routine PUTC provoque l'effacement d'une ligne d'écran. La ligne effacée sera la ligne courante du curseur, de même l'effacement se fera à partir de la colonne courante du curseur.

```
        .
        .
PUTC    EQU     $E803
        LDB     #$18
        JSR     PUTC
        .
        .
```


Affichages particuliers

Caractères accentués, alphabet G2

Le code ACC ($16) appliqué à la routine PUTC permet l'affichage d'une minuscule accentuée ou d'un caractère de l'alphabet G2. Dans le cas d'une minuscule accentuée, la procédure demande trois appels à PUTC :

1e appel : envoi du code ACC ($16)
2e appel : envoi du code d'accent
3e appel : envoi du code de la minuscule

Les accents disponibles sont les suivants:

Code	Accent ou signe correspondant
$41	Aigu
$42	Grave
$43	Circonflexe
$48	Tréma
$4B	Cédille

Nous rappelons que les minuscules sont accessibles par les codes compris entre $61 (a) et $7A (z), voir à ce sujet l'affichage des caractères alphanumériques, page 163.

Exemple: Pour afficher un ç nous écrirons:

```
        .
        .
PUTC    EQU     $E803
        LDB     #$16    Code ACC
        JSR     PUTC    1e appel
        LDB     #$4B    code cédille
        JSR     PUTC    2e appel
        LDB     #$63    code de c
        JSR     PUTC
        .
        .
```

D'une manière générale, si le dernier code envoyé ne correspond pas à une minuscule mais à un autre caractère (majuscule ou autre), ce dernier sera néanmoins affiché, et l'accent supprimé automatiquement.

Pour l'affichage d'un caractère de l'alphabet G2, la procédure s'écrit en deux appels:

1e appel: envoi du code ACC ($16)
2e appel: envoi du code caractère

Dans ce cas, le code caractère est à choisir parmi les suivants:

Code	Caractère affiché
$23	livre sterling
$24	dollar
$26	dièse
$27	paragraphe
$2C	flèche à gauche
$2D	flèche en haut
$2E	flèche à droite
$2F	flèche en bas
$30	degré
$31	plus ou moins
$38	division entière
$3C	un quart
$3D	un demi
$3E	trois quart
$6A	o dans e majuscule
$7A	o dans e minuscule
$7B	sz allemand

Le programme suivant affiche tous ces caractères:

```
* AFFICHAGE DES CARACTERES ALPHABET G2
* PAR CODE ACC ROUTINE PUTC

        TITLE   GENG2
        ORG     $A000
PUTC    EQU     $E803
        LDA     #$23
ENCOR   LDB     #$16
        JSR     PUTC
        TFR     A,B
        JSR     PUTC
        INCA
        CMPA    #$7C
        BNE     ENCOR
        SWI
        END
```


Caractères Télétel

Les caractères semi-grahiques aux normes Télétel sont également disponibles dans les TO8, TO9 et TO9+. Vous accèderez à ce générateur spécial de la manière suivante:

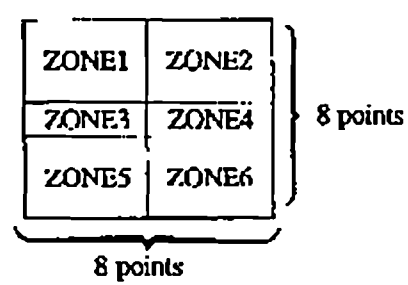
1e appel à PUTC, envoi du code $0E (selection mode Télétel)
2e appel à PUTC, envoi du code caractère

Le code caractère est compris entre les valeurs $20 à $7F, soit 64 caractères au total. Le programme suivant permet de les afficher à l'écran.

```
* AFFICHAGE DES CARACTERES TELETEL
* ET RETOUR AU MODE NORMAL


        TITLE   TELETEL
        ORG     $A000
PUTC    EQU     $E803
        LDB     #$0E        MODE TELETEL
        JSR     PUTC
        LDB     #$20
SUITE   JSR     PUTC
        INCB
        CMPB    #$80
        BNE     SUITE
        LDB     #$0F        MODE NORMAL
        JSR     PUTC
        SWI
        END
```

Notez que l'émission du code $0F permet de revenir au mode normal et, par conséquent, d'accéder à l'alphabet G0 par les codes compris entre $20 et $7F. Le principe de base de l'affichage Télétel est de ne plus considérer un caractère par une matrice de 8 × 8 points (comme l'alphabet G0 ,G2, utilisateur), mais par un ensemble de 6 zones réparties de la manière suivante:

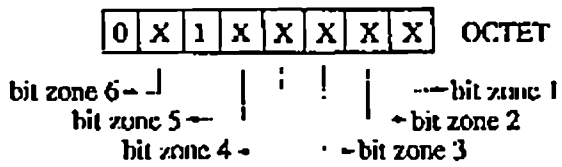


Le caractère semi-graphique est fabriqué en allumant une ou plusieurs zones. Nous comprenons ainsi que le nombre de combinaisons possibles nous limite à $2^6 = 64$ représentations différentes.

Chaque zone est contrôlée par un seul bit mémorisé:

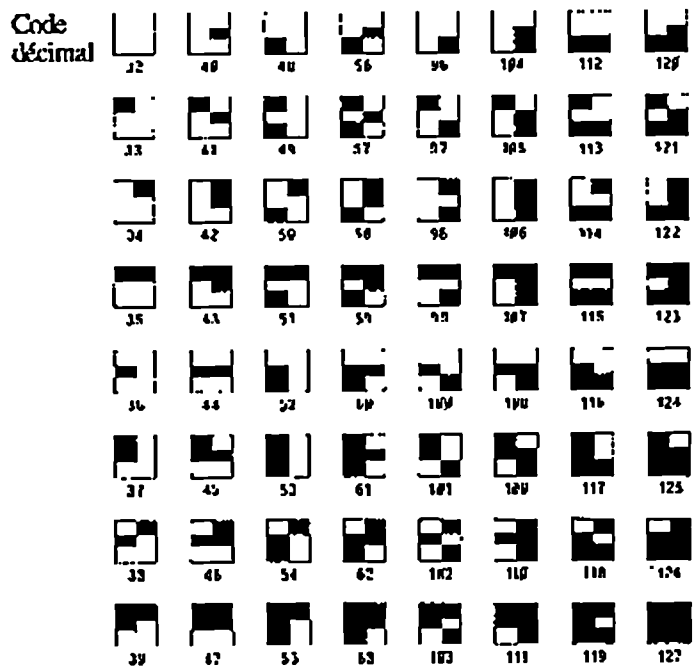
si le bit = 1, la zone est allumée, elle prendra la couleur de forme
si le bit = 0, la zone est éteinte, elle prendra la couleur de fond.

Ainsi un caractère Télétel est mémorisé par un seul octet de mémoire:



Il est bon de noter que le MSB et le 6e LSB doivent toujours respectivement rester à 0 et 1. L'ensemble des caractères pré-définis sont rassemblés dans le tableau suivant:

Caractères semi-graphiques Télétel



Séquences d'échappement

Les séquences d'échappement permettent de réaliser plusieurs tâches différentes, comme par exemple la modification des attributs de couleurs, le passage en mode d'incrustation vidéo ou le changement de la taille des caractères... Vous devrez utiliser deux méthodes d'appels différents à la routine PUTC ($E803) selon le travail désiré. Nous devons en effet distinguer deux modes de traitement possibles:

– le mode courant
– le mode plein écran.

Pour bien comprendre la différence, prenons un exemple: vous désirez écrire des caractères en rouge sur l'écran. Si cette couleur n'est utilisée que ponctuellement pour afficher un texte ou une portion de texte, coloyant des caractères déjà affichés à l'écran (ou à venir) dans une couleur différente, vous utiliserez le mode courant. Par contre, si l'ensemble de tout ce qui est (ou sera) affiché à l'écran doit être écrit en rouge, le mode plein écran sera utilisé. En langage BASIC, la différence est faite par le choix des instructions COLOR ou SCREEN.

Pour appeler une séquence d'échappement en mode courant, la procédure s'écrit en deux appels à la routine PUTC:

1e appel : envoi du code d'échappement ($1B)
2e appel : envoi du code réalisant la fonction desirée.

Exemple:

```
        .
        .
PUTC    EQU   $E803
        LDB   #$1B    code d'échappement
        JSR   PUTC
        LDB   #$XX    XX = code fonction
        JSR   PUTC
        .
        .
```

Par contre, en mode plein écran, la procédure nécessite 4 appels à la routine PUTC:

1e appel: envoi du code d'échappement $1B
2e appel: envoi du code $23
3e appel: envoi du code $20
4e appel: envoi du code réalisant la fonction desirée.

Exemple:

```
        .
        .
PUTC    EQU     $E803
        LDB     #$1B
        JSR     PUTC
        LDB     #$23
        JSR     PUTC
        LDB     #$20
        JSR     PUTC
        LDB     #$XX    XX = code fonction
        JSR     PUTC
          .
          .
```

Programmation des couleurs

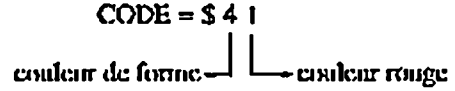
La séquence d'échappement permet de modifier les attributs de couleurs (forme, fond, tour). La valeur du code appelé XX dans les exemples précédents est définie de la manière suivante:
le digit de poids fort précise le destinataire du changement de couleur, il peut donc prendre trois valeurs possibles,

4 pour la forme
5 pour le fond
6 pour le tour.

Le digit de poids faible indique la couleur, selon la codification suivante:

0 pour le	Noir
1	Rouge
2	Vert
3	Jaune
4	Bleu
5	Magenta
6	Cyan
7	Blanc

Pour reprendre l'exemple cité plus haut, nous savons maintenant que pour écrire en rouge sur l'écran le code sera égal à:



Ainsi le programme suivant passera tous les caractères de la fenêtre de travail en rouge (mode plein écran).


```
        .
        .
PUTC    EQU     $E803
        LDB     #$1B    code d'échappement
        JSR     PUTC
        LDB     #$23    code intermédiaire
        JSR     PUTC
        LDB     #$20    code intermédiaire
        JSR     PUTC
        LDB     #$41    code couleur
        JSR     PUTC
          .
          .
```

Alors que le programme suivant écrira le message "OK" en rouge et vert sans modifier le reste de l'écran (mode courant).

```
* CE PROG AFFICHE OK EN ROUGE ET VERT
* ECHAPPEMENT DE TYPE COURANT

        TITLE   OK
        ORG     $A000
PUTC    EQU     $E803
        LDB     #$1B    ECHAPPEMENT
        JSR     PUTC
        LDB     #$41    FORME-ROUGE
        JSR     PUTC
        LDB     #$4F    AFFICHE O
        JSR     PUTC
        LDB     #$1B    ECHAPPEMENT
        JSR     PUTC
        LDB     #$42    FORME-VERT
        JSR     PUTC
        LDB     #$4B    AFFICHE K
        JSR     PUTC
        SWI
        END
```

En conclusion, la valeur du code peut être comprise entre

$40 à $47 pour la forme
$50 à $57 pour le tour
$60 à $67 pour le tour

Mais, comme vous l'avez déja certainement remarqué (petits futés), nous ne travaillons dans ce cas que sur 8 couleurs. La table de valeurs a donc été étendue de $70 à $87 afin d'accéder aux 8 couleurs pastel et, dans ce cas, les allocations sont les suivantes :

$70 à $77 pour la forme
$78 à $7F pour le fond
$80 à $87 pour le tour

avec

0 pour le Gris
1 Rose
2 Vert clair
3 Sable
4 Bleu clair
5 Parme
6 Bleu ciel
7 Orange

Programmation des modes d'affichage

Comme nous l'avons détaillé dans les études matérielles, les TO8, TO9 et TO9+ possèdent de nombreux modes d'affichage différents, permettant de faire ponctuellement des compromis entre couleurs, définition graphique et rapidité de visualisation. L'accès de ces modes nécessite 2 appels à la routine PUTC:

1e appel: envoi du code d'échappement ($1B)
2e appel: envoi du code de mode.

Le code de mode desiré sera l'un des suivants:

Code	Mode sélectionné
$48	Page 1
$49	Page 2
$4A	Superposition écriture page 2
$4B	Superposition écriture page 1
$59	Bit-map 4 couleurs
$5A	40 colonnes
$5B	80 colonnes
$5E	Bitmap 16 couleurs
$6D	Incrustation
$6C	Incrustation (off)
$88	Triple superposition sélection page 1
$89	Triple superposition sélection page 2
$8A	Triple superposition sélection page 3
$8B	Triple superposition sélection page 4

Exemple: pour travailler en mode 80 colonnes, nous utiliserons la procédure suivante:

```
        .
        .
PUTC    EQU     $E803
        LDB     #$1B    code d'échappement
        JSR     PUTC
        LDB     #$5B    mode 80 colonnes
        JSR     PUTC
        .
        .
```

Si l'unité centrale est équipée de l'interface d'incrustation (en option), vous pourrez alors la commuter dans ce mode spécial de visualisation, en envoyant le code $6D. L'incrustation permet de mélanger une image vidéo issue du téléviseur et l'image synthétique créée par le micro-ordinateur. La coexistence de ces deux images simultanées à l'écran est également fonction de la programmation du circuit PALETTE, detaillée page 212.

Nous vous conseillons de vous reporter à la partie de ce livre traitant du fonctionnement des TO8, TO9 et TO9+ si vous désirez de plus amples détails sur les particularités des modes Page, Bit-map, etc.

Dimensions des caractères

Les caractères alphanumériques peuvent être affichés à l'écran selon diverses dimensions repertoriées ci-dessous.

La déclaration se réalise par deux appels à la routine PUTC:

1e appel : envoi du code d'échappement $1B (ACCU B)
2e appel : envoi du code de dimension (ACCU B)

Code	Dimensions des caractères
$4C	Taille normale
$4D	Double hauteur, largeur normale
$4E	Double largeur, hauteur normale
$4F	Double taille

Exemple: pour travailler en double taille, nous écrirons le programme suivant:

```
        ORG   $A000
PUTC    EQU   $E803
        LDB   #$1B   code d'échappement
        JSR   PUTC   premier appel
        LDB   #$4F   code double taille
        JSR   PUTC   second appel
        SWI
        END
```

Traitements divers

Certains traitements particuliers repertoriés ci-dessous sont également définis sous le contrôle de séquence d'échappement:

Code	Traitement correspondant
$58	Masquage
$5F	Démasquage
$5C	Inversion de la vidéo
$6A	Scroll normal
$6E	Scroll doux
$6B	Mode page (pas de scroll)
$68	Ecriture d'un caractère sans modifier la couleur
$69	Ecriture d'un caractère dans la couleur courante

La procédure s'établit en deux appels de PUTC:

1e appel : envoi du code d'échappement $1B
2e appel : envoi du code représentant le traitement desiré.

Exemple: pour avoir un listing d'éditeur défilant doucement à l'écran, nous pouvons écrire le programme suivant:

```
        ORG   $A000
PUTC    EQU   $E803
        LDB   #$1B
        JSR   PUTC
        LDB   #$6E
        JSR   PUTC
        SWI
        END
```

Vous pourrez constater ce scroll doux en faisant des dumpings sous monitor. Notons que certains codes, comme par exemple $5C pour l'inversion vidéo, peuvent être appelés en mode plein écran selon la procédure décrite en début de ce chapitre. Le masquage consiste à écrire des caractères en couleur noire sur fond noir, jusqu'à ce que l'attribut de démasquage soit sollicité. En utilisant ce dernier selon le mode plein écran, vous dévoilerez d'un seul coup tous les caractères écrits en mode masqué.

Affichage alphanumérique par la routine PLOT

La routine PLOT, utilisée plus fréquemment pour afficher des points graphiques (voir page 183), peut également être sollicitée pour l'affichage de caractères alphanumériques. Dans ce cas, le code ASCII du caractère à afficher est implanté dans le registre CHDRAW ($6041), la couleur est precisée dans le registre COLOUR ($603B), et ses coordonnées sont exprimées dans les registres X et Y du 6809 E (représentant respectivement l'abscisse et l'ordonnée).

Les coordonnées dans X sont comprises entre 1 et 40 décimal, pour le mode 40 colonnes, ou entre 1 et 80 pour le mode 80 colonnes. Les valeurs de Y sont comprises entre 0 et 24 décimal pour les deux modes.

Les attributs de couleurs sont identiques à ceux exprimés page 181. L'octet envoyé à COLOUR est décomposable en deux digits, le digit de poids faible code la couleur de fond, le digit de poids fort code la couleur de forme. En retour de la routine PLOT, les valeurs implantées dans les registres X et Y sont automatiquement remplies dans les registres PLOTX et PLOTY.

Nom.	: PLOT
Adresse du point d'entrée	: $E80F
Paramètre d'entrée.	: registres X et Y du 6809 E registre CHDRAW $6041 registre COLOUR $603B
Paramètre de retour	: registre PLOTX $603D-$603E registre PLOTY $603F-$6040.
Effet	: affiche à l'écran un caractère exprimé en ASCII dans CHDRAW, aux coordonnées passées par X et Y, et dans la couleur définie par COLOUR
Exemple	: le programme suivant affiche le caractère "K" au milieu de l'écran, en noir sur fond blanc.

```
* AFFICHAGE D'UN CARACTERE PAR LA
* LA ROUTINE PLOT

        TITLE   PLOTCAR
        ORG     $A000
PLOT    EQU     $E80F
CHDRAW  EQU     $6041
COLOUR  EQU     $603B
        LDA     #$4B        CODE ASCII DE 'K'
        STA     CHDRAW
        LDA     #$40        COULEUR NOIR/BLANC
        STA     COLOUR
        LDX     #$0014      COLONNE 20
        LDY     #$006C      LIGNE 12
        JSR     PLOT
        SWI
        END
```

Note: On peut accéder directement à l'écriture du caractère repéré dans CHDRAW en faisant un appel à l'adresse $E833 (CHPL). L'accès au mode Bit-map 16 et Triple superposition sont interdits dans ce mode de PLOT.

3. Gestion graphique de l'écran

Mémorisation en RAM forme et couleur

Comme nous l'avons expliqué dans l'étude matérielle du TO9, la RAM écran est en fait constituée d'une RAM A et d'une RAM B dont le fonctionnement ainsi que les octets mémorisés sont fonction du type de visualisation utilisé (mode 40 ou 80 colonnes, bit-map, etc.).

Néanmoins, dans le mode commun au TO7/70 (40 colonnes, 16 couleurs, 320 × 200 points), la RAM A correspond à la mémoire forme et la RAM B à la mémoire couleur.

Ces deux RAM sont décodées aux mêmes adresses comprises entre $4000 et $5FFF et la sélection de l'une d'entre elles est effectuée par le bit forme issu du PIA interne au 6846 ($E7C3).

Pour les TO8 et TO9+, les notions sont les mêmes bien que physiquement rien ne soit comparable. La différence fondamentale est liée à l'utilisation du gate mode page qui intègre, grosso modo, les fonctions des gates de gestion machine et gate d'affichage dans un même boîtier.

D'autre part, au niveau de l'organisation mémoire nous passons d'une structure de commutation de banques par commutation de boîtiers RAM (TO9), à une commutation de banques par commutation de pages dans un même boîtier RAM (TO8 et TO9+).

Que deviennent dans ces conditions le bit de forme à l'adresse $E7C3 et les notions de RAM écran forme et couleur ? Disons tout simplement que le constructeur a eu la bonne idée d'émuler ces mêmes principes dans les TO8 et TO9+. Même si du côté matériel tout est différent, du côté logiciel tout est compatible. Nous pourrons donc continuer à "poker" aveuglément l'adresse $E7C3 bien que le bit de forme n'existe plus ni physiquement ni à cette adresse! Miraculeux, non ?

Commutation couleur

Pour travailler dans la mémoire couleur, il suffit de forcer à 0 le bit 0 de l'adresse $E7C3 (forme). Il est conseillé d'utiliser la technique de programmation par "masques" pour ne modifier que la valeur de ce bit.

Commutation forme

Inversement, pour travailler dans la mémoire forme, il faut forcer à 1 le bit 0 de l'adresse SE7C3 (à l'exclusion d'autres bits).

Allumage ou extinction d'un point graphique

La routine PLOT présentée ci-dessous, permet de changer la couleur d'un point graphique dont les coordonnées sont exprimées dans le registre X (abscisses) et le registre Y (ordonnées). Si l'ordonnée est toujours comprise entre 0 et 199, l'intervalle des abscisses dépend du mode d'affichage choisi:

Valeurs limites pour X		Mode graphique:	
Décimal	Hexadécimal		
0-319	$0-$13F	TO7	16 couleurs
0-319	$0-$13F	Page 1	2 couleurs
0-319	$0-$13F	Page 2	2 couleurs
0-319	$0-$13F	Bit-map	4 couleurs
0-319	$0-$13F	Superposition	
0-639	$0-$27F	80 colonnes	2 couleurs
0-159	$0-$9F	Bit-map	16 couleurs

La couleur du point sera precisée dans le registre FORME ($6038), les valeurs comprises entre -16 et +15 seront interprétées de la manière suivante:

Couleur	Code forme	Code fond
Noir	0	-1
Rouge	1	-2
Vert	2	-3
Jaune	3	-4
Bleu	4	-5
Magenta	5	-6
Cyan	6	-7
Blanc	7	-8
Gris	8	-9
Rose	9	-10
Vert clair	10	-11
Sable	11	-12
Bleu clair	12	-13
Parme	13	-14
Bleu ciel	14	-15
Orange	15	-16

```
Nom                    : PLOT
Adresse du point d'entrée : $E80F
Paramètres d'entrée    : Registre X et Y du 6809 E
                         Registre FORME $6038
                         Registre CHDRAW $6041
                         Registre STATUS $6019
Paramètres de retour   : Registre PLOTX $603D-$603E
                         Registre PLOTY $603F-$6040
Effet                  : Affiche un point graphique dont les coordonnées
                         sont precisées dans X et Y du 6809 E, et la couleur
                         exprimée dans le registre FORME.
Exemple

* AFFICHAGE D'UN POINT GRAPHIQUE

        TITLE   POINT
        ORG     $A000
PLOT    EQU     $E80F
CHDRAW  EQU     $6041
STATUS  EQU     $6019
FORME   EQU     $6038
        LDA     STATUS
        ANDA    #$EF        BIT4 FORCE A 0
        STA     STATUS
        CLR     CHDRAW      PLOT EN MEST GRAPH
        LDA     #$01        COUL ROUGE
        STA     FORME
        LDX     #$013F      DERNIERE COLONNE
        LDY     #$0064      LIGNE=100
        JSR     PLOT
        SWI
        END
```

En conséquence, il est bon de noter que les valeurs positives appellent des couleurs de forme et inversement, les codes négatifs sélectionnent des couleurs de fond. Un code de couleur fond se déduit d'un code de forme en ajoutant 1 et en prenant l'opposé.

Précisons quelques remarques importantes relatives aux modes utilisés. En mode TO7, si le code de la couleur est négatif, le point sera écrit en fond, cela implique que le bit correspondant dans l'octet de mémoire forme soit mis à 0. Inversement, si le code couleur est positif, le point sera écrit en forme, le bit correspondant dans l'octet de mémoire forme sera mis à 1.

Si le bit 4 du registre STATUS ($6019) est à 1, seul le bit en mémoire forme sera modifié, la couleur ne sera pas changée. En mode Page1, Page2, Superposition et Triple superposition, un code positif écrit dans la couleur de la page sélectionnée, un code négatif écrit dans la couleur fond.

Pour le mode bit-map 4 couleurs, seuls les 4 premiers codes positifs sont utilisables (de 0 à 3) et travaillent dans la mémoire forme, le fond n'a aucune signification.

Pareillement, le mode bit-map 16 couleurs n'autorise que les 16 codes positifs (0 à 15) en forme, le fond n'ayant aucun sens. Le mode 80 colonnes ne comprend à la fois qu'un code positif écrit en forme et un code négatif écrit en fond.

Enfin, dans tous les modes choisis, le registre CHDRAW ($6041) doit être mis à 0 pour avoir une gestion graphique de l'écran. Si CHDRAW est différent de 0, la routine PLOT sera utilisée en mode alphanumérique, décrit page 181.

En retour de la routine PLOT, les coordonnées du dernier point affiché seront automatiquement recopiées des registres X et Y (6809 E) dans les registres PLOTX ($603D-$603E) et PLOTY ($603F-$6040).

Tracé d'un segment de droite

La routine DRAW trace un segment de droite entre deux points déterminés que nous appellerons origine et destination. Les coordonnées du point d'origine (abscisse et ordonnée) sont définis respectivement par le contenu des registres PLOTX ($603D-$603E) et PLOTY ($603F-$6040). Les coordonnées du point destination (abscisse et ordonnée) sont définis respectivements par le contenu des registres X et Y du 6809 E.

Les valeurs limites de ces coordonnées sont identiques à celles exprimées précédemment (page 183). La couleur du segment tracé est définie par le registre FORME selon les mêmes conventions que ci-dessus. Le contenu du registre CHDRAW doit être nul, sinon la routine DRAW travaillera en mode caractère (voir page 188).

En retour de la routine DRAW, les coordonnées du dernier point tracé sont recopiées automatiquement dans les registres PLOTX et PLOTY. Cette particularité simplifie énormément l'écriture de programmes pour dessiner des figures géométriques tels que rectangles, carrés, triangles, losanges... Car après le tracé du premier segment, seuls les coordonnées du point destination sont à préciser pour tracer les suivants.


```
Nom                 : DRAW
Adresse du point d'entrée : $E80C
Paramètres d'entrée : Registre X et Y du 6809 E
                      Registre PLOTX $603D-$603E
                      Registre PLOTY $603F-$6040
                      Registre CHDRAW $6041
                      Registre FORME $6038
                      Registre STATUS $6019
                      Registre COLOUR $6FEB
Paramètre de retour : Registre PLOTX $603D-$603E
                      Registre PLOTY $603F-$6040
Effet               : Tracé un segment de droite
Exemple             : Le programme ci-dessous utilise quatre fois la
                      routine DRAW pour dessiner un losange.

* DESSIN D'UN LOSANGE BLEU PAR LA
* ROUTINE DRAW

        TITLE   LOSANGE
        ORG     $A000
DRAW    EQU     $E80C
CHDRAW  EQU     $6041
STATUS  EQU     $6019
FORME   EQU     $6038
PLOTX   EQU     $603D
PLOTY   EQU     $603F

        LDA     STATUS
        ANDA    #$EF        BIT4 FORCE A 0
        STA     STATUS
        CLR     CHDRAW      DRAW EN GEST GRAPH
        LDA     #$04        COUL BLEU
        STA     FORME
        LDX     #$00A0      COLONNE 160
        LDY     #$0040      LIGNE   64
        STX     PLOTX
        STY     PLOTY
        LDX     #$00D0      COLONNE 208
        LDY     #$0070      LIGNE   112
        JSR     DRAW        TRACE SEGMENT 1
        LDX     #$00A0      COLONNE 160
        LDY     #$00A0      LIGNE   160
        JSR     DRAW        TRACE SEGMENT 2
```

```
        LDX     #$0070  COLONNE 112
        LDY     #$0070  LIGNE   112
        JSR     DRAW    TRACE SEGMENT 3
        LDX     #$00A0  COLONNE 160
        LDY     #$0040  LIGNE   64
        JSR     DRAW    TRACE SEGMENT 4
        SWI
        END
```

Note: Pour modifier les couleurs, le bit 4 de STATUS ($6019) doit être forcé à 0.

Dessiner avec des caractères

L'intérêt de DRAW est de pouvoir travailler également avec des caractères au lieu de points grahiques. Dans ce cas, le registre CHDRAW doit contenir le code ASCII du caractère qui nous servira à dessiner. La couleur est determinée par le registre COLOUR, le digit de poids fort définissant la couleur de fond, le digit de poids faible déterminant la couleur de forme.

Les coordonnées passées par PLOTX, PLOTY, X et Y du 6809 E sont comprises entre 0 et 24 décimal pour les ordonnées, les abscisses dépendant du mode de visualisation:

1 à 40 pour le mode 40 colonnes
1 à 80 pour le mode 80 colonnes.

Ainsi, nous pouvons reprendre le programme précédent dont la structure est la même. En modifiant simplement le contenu de CHDRAW, les valeurs des coordonnées et en initialisant le registre COLOUR, nous traçons le même losange mais avec les lettres "K".


```
* DESSIN D'UN LOSANGE CONSTITUE DE 'K'

        TITLE   LOSANGK
        ORG     $A000
DRAW    EQU     $E80C
CHDRAW  EQU     $6041
STATUS  EQU     $6019
PLOTX   EQU     $603D
PLOTY   EQU     $603F
COLOUR  EQU     $603E


        LDA     STATUS
        ANDA    #$EF    BIT4 FORCE A 0
        STA     STATUS
        LDA     #$4B    CODE ASCII DE 'K'
        STA     CHDRAW  DRAW EN GEST CARAC
        LDA     #$40    COUL NOIR/BLANC
        STA     COLOUR
        LDX     #$0014  COLONNE  20
        LDY     #$0008  LIGNE    08
        STX     PLOTX
        STY     PLOTY
        LDX     #$001A  COLONNE  26
        LDY     #$000E  LIGNE    14
        JSR     DRAW    TRACE SEGMENT 1
        LDX     #$0014  COLONNE  20
        LDY     #$0014  LIGNE    20
        JSR     DRAW    TRACE SEGMENT 2
        LDX     #$000E  COLONNE  14
        LDY     #$000E  LIGNE    14
        JSR     DRAW    TRACE SEGMENT 3
        LDX     #$0014  COLONNE  20
        LDY     #$0008  LIGNE    08
        JSR     DRAW    TRACE SEGMENT 4
        SWI
        END
```

4. Lecture de l'écran

Lecture d'un point graphique

Le principe général est de viser un point à l'écran qui sera repéré par ses coordonnées (registres X et Y du 6809 E), afin de lire sa couleur (retour dans l'accu B). L'utilisation des valeurs et l'interprétation des résultats de la routine GETP réalisant cette tâche sont fonction du mode d'affichage selectionné. Afin de mieux vous repérer, nous avons répertoriés dans le tableau ci-dessous les divers cas possibles:

Mode d'affichage	Coordonnées du point Ordonnée Y	Abscisse X	Valeur lue dans B
40 colonnes (TO7)	0 à 199	0 à 319	-16 à -1 si coul. fond 0 à +15 si coul. forme
Bit-map 4 couleurs Page 1 Page 2 Superposition	0 à 199	0 à 319	0 à 3 0 si point en forme -1 si point en fond
Bit-map 16 couleurs	0 à 199	0 à 159	0 à +15
80 colonnes	0 à 199	0 à 639	0 si point en forme -1 si point en fond

La numérotation des couleurs renvoyées par GETP est identique à la codification énoncée page 184.

```
Nom                    : GETP
Adresse du point d'entrée : $E82I
Paramètre d'entrée     : Registre X et Y du 6809 E
Paramètre de retour    : Accumulateur B du 6809 E
Effet                  : Retourne dans B la couleur du point visé par X et Y
Exemple                : En supposant que vous travaillez avec la cartouche
                         ASSEMBLEUR TOTEK, sous editor nous avons
                         un bandeau rouge reservé aux messages d'erreurs ou
                         de confirmation.
```

Le programme ci-dessous va lire un point correspondant à ce bandeau. En retour nous aurons la valeur $FE dans l'accumulateur B du 6809 E ($FE = - 02 couleur de fond rouge).

```
        ORG   $A000
GETP    EQU   $E82I
        LDX   #$013E   colonne 318
        LDY   #$00C7   ligne 199
        JSR   GETP
        SWI
        END
```


Lecture d'un caractère

La routine GETS peut être utilisée dans les modes TO7, 80 colonnes, Page 1, Page 2, Superposition. Vous ne devez pas l'appeler dans les modes Bit-map 4 couleurs, Bit-map 16 couleurs et Triple superposition. GETS retourne dans l'accumulateur B le code ASCII du caractère dont les coordonnées sont précisées dans le registre X (abscisse) et l'accumulateur A (ordonnée).

$0 \le A \le 24$ et $1 \le X \le 80$ pour le mode 80 colonnes
$0 \le A \le 24$ et $1 \le X \le 40$ pour le mode 40 colonnes

Si, à l'endroit visé le caractère n'est pas connu, la routine GETS renverra la valeur 0 dans l'accumulateur B. Sinon trois cas peuvent se produire, nous vous poposons de les étudier.

Caractère normal

Le caractère appartient à l'alphabet G0. GETS retourne dans l'accumulateur B le code ASCII correspondant.

Minuscule accentuée ou c cédille

Le caractère visé est une minuscule accentuée, dans ce cas GETS retourne dans B le code de l'accentuation ACC soit $16, et il est nécessaire de faire deux autres appels à GETS:

2e appel : B retourne le code de l'accent
3e appel : B retourne le code ASCII de la minuscule.

Caractère de l'alphabet G2

Si le caractère visé appartient à l'alphabet G2, le code ACC ($16) sera retourné dans B. I faudra alors faire un 2e appel à GETS pour lire le code du caractère:

2e appel : B retourne le code du caractère.

Nom	: GETS
Adresse du point d'entrée	: $E824
Paramètre d'entrée	: Accu A et registre X du 6809 E
Paramètre de retour	: Accu B du 6809E
Effet	: retourne le code ASCII du caractère défini par ces coordonnées d'écran (A = ordonnée, X = abscisse)
Exemple	: Si vous travaillez avec la cartouche ASSEMBLEUR TOTEK, sous moniteur un bandeau jaune sur le haut de la fenêtre de travail sert à fabuler l'affichage du contenu des registres du microprocesseur 6809 E.

Le programme ci-dessous lit le caractère P de ce bandeau et retourne en conséquence le code $50 dans B.

```
        ORG   $A000
GETS    EQU   $E824
        CLRA          1e ligne
        LDX   #$0002  2e colonne
        JSR   GETS
        SWI
        END
```


5. Gestion du clavier

Lecture rapide du clavier

La routine KTST fait un balayage rapide des touches du clavier pour détecter si l'une d'entre elles est appuyée. Le résultat de ce test est consigné dans le BIT C du registre d'état du microprocesseur:

si C = 0 aucune touche n'est appuyée
si C = 1 une touche est enfoncée.

Aucune reconnaissance de la touche appuyée n'étant effectuée par KTST, l'exécution de cette routine est très rapide.

Nom	: KTST
Adresse du point d'entrée	: $E809
Paramètre d'entrée	: Néant
Paramètre de retour	: BIT C du registre d'état du 6809 E
Effet	: Positionne C à 1 si une touche est appuyée.
Exemple	:

```
* LECTURE RAPIDE DU CLAVIER.
* LA PRESSION D'UNE TOUCHE ENTRAINE SWI.

        TITLE   TESTCLAV
        ORG     $A000
KTST    EQU     $E809
        ANDCC   #$FE      FORCE C A 0
        LDX     #$FFFF
ENCOR   LEAX    -1,X      BOUCL INHIB CLAV
        BNE     ENCOR
SUITE   JSR     KTST
        BCC     SUITE     BRANCH SI C=0
        SWI
        END
```

Décodage du clavier

La routine GETC est plus complète que la précédente (KTST). En effet, elle teste le clavier pour détecter une touche éventuellement appuyée, mais elle identifie également cette touche et retourne son code ASCII dans l'accumulateur B du microprocesseur. Etant donné le nombre de touches constituant le clavier et leurs fonctions parfois doubles, plusieurs cas peuvent se présenter:

• si la touche est celle d'un caractère simple de l'alphabet G0, son code ASCII sera alors retourné dans l'accu B au premier appel de GETC;

• si aucune touche n'est enfoncée ou s'il s'agit de SHIFT ou CNT appuyée seule, ou encore si plusieurs touches parmi SHIFT ou CNT ont été enfoncées simultanément, la valeur 0 sera retournée dans l'accu B;

• la touche SHIFT sélectionne le caractère se trouvant en haut de la touche, le code ASCII du caractère ainsi sélecté sera envoyé dans B;

• la touche CNT force à 0 le bit 6 du code ASCII du caractère appuyé simultanément;

• s'il s'agit d'un caractère accentué et quelle que soit la procédure de saisie, la lecture se fera en trois appels de GETC:

– 1e appel retourne dans B le code ACC (S16)
– 2e appel retourne dans B le code de l'accent ou de la cédille
– 3e appel retourne dans B le code de la minuscule.

Comme nous l'avons étudié dans l'analyse matérielle, la chaîne de traitement du clavier envoie une interruption IRQ au microprocesseur à chaque fois qu'une touche est appuyée. Lors de la réception de ce caractère, le microprocesseur le range dans un buffer circulaire dont l'adresse est située dans le registre BUFCLV ($6079-$607A).

Notez que ce registre étant en RAM, l'utilisateur peut resituer ce buffer à l'endroit de son choix. Au démarrage, ce buffer est positionné dans la page 0 du moniteur et possède une capacité de 5 caractères. Quand il est plein, les caractères suivants sont ignorés! Précisément, le registre SIZCLV ($607B) permet de déclarer la taille de ce buffer; on peut ainsi augmenter (ou diminuer) sa capacité. Mais vous noterez ces valeurs limites:

– SIZCLV étant un registre de 8 bits, la valeur maximum ($FF) correspond à un buffer de 255 caractères au plus.
– la taille minimum à préserver est de 3 caractères pour garantir la lecture des caractères accentués.

Les TO8, TO9 et TO9+ disposent de 10 touches de fonctions. Les codes envoyés par ces touches sont compris entre $90 et $99 (inclus). Les pavés numériques peuvent être reconfigurés de façon à envoyer des codes différents de ceux inhérents aux touches. Dans ce cas, les 10 chiffres de 0 à 9 envoient les codes de $9A à $A3, le point envoie le code $A4 et la touche ENT le code $A5.

```
Nom                 : GETC
Adresse du point d'entrée : $E806
Paramètre d'entrée  : Registre BUFCLV $6079-$607A
                      Registre SIZCLV $607B
Paramètre de retour : Bit C du RE 6809 E
                      ACCU B du 6809 E
Effet               : Réalise une lecture du clavier. Si une touche est
                      appuyée:
                        - le bit C du RE passe à 1
                        - l'accu B retourne le code ASCII
Exemple

* LECTURE DU CLAVIER. LA PRESSION D'UNE
* TOUCHE PROVOQUE SWI, LIRE CODE ASCII
* DANS L'ACCU B PAR COMMANDE R

        TITLE   CLAVIER1
        ORG     $A000
GETC    EQU     $E806
ENCORE  JSR     GETC
        BCC     ENCORE
        SWI
        END
```

Dans ce second exemple, nous avons chaîné GETC et PUTC. En effet, cette opération se réalise facilement car le paramètre de sortie de GETC correspond au paramètre d'entrée de PUTC (Accu B, code ASCII). Ainsi le programme ci-dessous affiche à l'écran le caractère tapé au clavier.

```
* LECTURE DU CLAVIER ET AFFICHAGE DU
* CARACTERE CORRESPONDANT A L'ECRAN
* UTILISATION DE GETC ET PUTC


        TITLE   CLAVIER2
        ORG     $A000
GETC    EQU     $E806
PUTC    EQU     $E803
ENCORE  JSR     GETC
        BCC     ENCORE  test bit C=1
        JSR     PUTC
        BRA     ENCORE
        END
```

Programmation du clavier

Le dialogue entre l'unité centrale et le clavier est bi-directionnel. Nous avons en effet la possibilité d'envoyer divers codes au clavier pour le programmer. C'est une autre utilisation de la routine GETC, employée en association du registre STATUS ($6019). Pour envoyer un code au clavier, il faut que le bit 1 de STATUS soit à 1. En retour de GETC, le bit 1 de STATUS est automatiquement remis à 0.

Nom : GETC
Adresse du point d'entrée : $E806
Paramètre d'entrée : Registre STATUS $6019
Accu B du 6809 E
Paramètre de retour : Néant
Effet : en fonction du code implanté dans B. Les effets sont les suivants:

Code dans B	Effet
$F8	RESET soft du clavier: - CAPSLOCK on - keypad selecté pour les chiffres - périphériques pouvant parler (TO9)
$F9	CAPS LOCK on
$FA	CAPS LOCK off
$FB	Sélection codes spéciaux pour le clavier
$FC	Sélection chiffres pour le clavier
$FD	Périphériques autorisés à émettre
$FE	Périphériques interdits d'émettre (TO9)

Exemple :

```
* CE PROG POSITIONNE LE CAPS LOCK OFF
* (VOYANT ROUGE ETEIND)


        TITLE   CLAVIER3
        ORG     $A000
GETC    EQU     $E806
STATUS  EQU     $6019
        LDB     STATUS
        ORB     #$02
        STB     STATUS
        LDB     #$FA
        JSR     GETC
        SWI
        END
```

Périphériques du clavier

Une prise 9 broches située sur le flanc arrière du clavier du TO9 permet la connexion de divers périphériques, par exemple la souris. Le bit 6 du registre CONFIG (S6074) indique la présence du périphérique clavier:

bit 6 – 0 pas de périphérique connecté
bit 6 = 1 périphérique connecté.

Le bit 7 du même registre (CONFIG) indique si la souris remplace ou non le light pen.

bit 7 = 0 light pen en fonctionnement
bit 7 = 1 souris remplace le light pen.

Dans le second cas, les appels à LPIN et GETL (voir page 200) sont détournés sur les routines PEIN et GEPE.

Test des boutons du périphérique

La routine PEIN teste l'état des boutons du périphérique connecté au clavier. La réponse est retournée dans le registre d'état du microprocesseur:

bit C = 1 bouton n°1 enfoncé
bit C = 0 cas contraire

bit Z = 1 bouton n°2 enfoncé
bit Z = 0 cas contraire.

A noter que, dans le cas de la souris, le bouton n°2 est le poussoir de droite.

Lecture du périphérique

La routine GEPE lit les coordonnées issues de la position du périphérique et retourne dans Y l'ordonnée (0 à 199 décimal) et dans X l'abscisse correspondante (selon les modes, 0 à 159, 0 à 319 ou 0 à 639). Si la mesure est correcte, le bit C du registre d'état est forcé à 0, dans le cas contraire il est forcé à 1.

```
Nom                    : PEIN
Adresse du point d'entrée : $EC09
Paramètre d'entrée     : Néant
Paramètre de retour    : Bit C et bit Z du RE 6809 E
Effet                  : Test des boutons du périphérique

Nom                    : GEPE
Adresse du point d'entrée : $EC06
Paramètre d'entrée     : Néant
Paramètre de retour    : Bit C du RE
                         Registre X et Y du 6809 E
Effet                  : lecture de la position du périphérique
Exemple :

* CE PROGRAMME PERMET DE DESSINER
* AVEC LA SOURIS. CLIQUER LE BOUTON
* DE GAUCHE ENTRAINE SWI


        TITLE   SOURIS
CONFIG  EQU     $E07A
GEPE    EQU     $EC06
PLOT    EQU     $E80F
FORME   EQU     $6038
PEIN    EQU     $EC09
```

```
        ORG     $A000
        LDA     CONFIG
        ORA     #$C0        bit 7 et 6 à 1
        STA     CONFIG      souris reconnue
        LDA     #$01        01=code coul rouge
        STA     FORME       coul forme rouge
ENCOR   JSR     GEPE
        JSR     PLOT
        JSR     PEIN
        BCC     ENCOR       test bout gauche
        SWI
        END
```

6. Gestion du light pen

Test du switch light pen

La routine LPIN permet de tester l'état du switch situé sur l'extrémité du light pen. La réponse est retournée dans le bit C (carry) du registre d'état du microprocesseur 6809 E:

Si C = 1, le switch est (ou a été) enfoncé
Si C = 0, le switch n'est pas enfoncé.

Le bit Z du registre d'état est toujours forcé à 0. Un anti rebond de 10 millisecondes est effectué automatiquement. Dans le cas où la souris est connectée, se reporter page 193.

Nom : LPIN
Adresse du point d'entrée : $E81B
Paramètre d'entrée : Néant
Paramètre de retour : Registre d'état du 6809 E
Effet : Positionne C à 1 si le switch est enfoncé
Exemple : Dans le programme ci-dessous, l'appui sur le switch entraîne un SWI.

```
        ORG   $A000
LPIN    EQU   $E81B
ENCOR   JSR   LPIN
        BCC   ENCOR   test C = 1
        SWI
        END
```

Lecture de la zone pointée

La routine GETL présentée ci dessous lit les coordonnées d'un point visé par le light pen et les retourne dans le registre X (abscisse) et Y (ordonnée) du 6809 E.

0 ≤ Y ≤ 199 et 0 ≤ X ≤ 159 pour le mode bit-map 16
0 ≤ Y ≤ 199 et 0 ≤ X ≤ 638 pour le mode 80 colonnes
0 ≤ Y ≤ 199 et 0 ≤ X ≤ 319 pour les autres modes.

En mode 80 colonnes, l'abscisse est obligatoirement paire. La mesure s'effectue sur une trame TV. En cas de problème de lecture (luminosité de l'écran trop faible, light pen trop éloigné de l'écran...), la routine GETL force le bit C du registre d'état du 6809 E à 1. Dans le cas où la souris est connectée, se reporter page 193.

Nom : GETL
Adresse du point d'entrée : $E818
Paramètre d'entrée : Néant
Paramètre de retour : Registre X et Y du 6809 E
bit C du RE du 6809 E
Effet : Lit le point visé par le light pen et retourne ses coordonnées dans X et Y.
Exemple :

```
* CE PROGRAMME PERMET DE DESSINNER
* SUR L'ECRAN AVEC LE LIGHT PEN EN BLEU
* SUR FOND BLANC. ON SORT DU PROGRAMME
* EN APPUYANT SUR LE SWITCH.

        TITLE   DESLIGHT
        ORG     $A000
GETL    EQU     $E818
PUTC    EQU     $E803
PLOT    EQU     $E80F
LPIN    EQU     $E81B
CHDRAW  EQU     $6041
STATUS  EQU     $6019
FORME   EQU     $6038
        LDA     STATUS
        ANDA    #$EF      BIT4 FORCE A 0
        STA     STATUS
        CLR     CHDRAW    PLOT EN GEST GRAPH
        LDA     #$04      COUL FORME  BLEU
        STA     FORME
        LDB     #$0C
        JSR     PUTC      EFFACE FENETRE
        LDB     #$1B      SEQUENC ECHAPP
        JSR     PUTC
        LDB     #$23      PLEIN-
        JSR     PUTC
        LDB     #$20      -ECRAN
        JSR     PUTC
        LDB     #$57      FOND BLANC
        JSR     PUTC
ENCORE  JSR     GETL      LECT LIGHTPEN
        JSR     PLOT      AFFICH POINT VISE
        JSR     LPIN      LECT DU SWITCH
        BCC     ENCORE
        SWI
        END
```

7. Gestion des manettes de jeux

La routine JOYS fait la lecture de la position d'une manette de jeux. L'accumulateur A du 6809 E contiendra le code de la manette à lire (0 ou 1), l'accumulateur B retourne sa position conformément au codage suivant:

Code renvoyé dans B	Position correspondante
$00	Centre
$01	Nord
$02	Nord-est
$03	Est
$04	Sud-est
$05	Sud
$06	Sud-ouest
$07	Ouest
$08	Nord-ouest

Le bit de retenue (C) du registre d'état 6809 E est mis à 1 si le bouton ACTION a été enfoncé, et à 0 dans le cas contraire. Il n'y a pas d'anti-rebond prévu pour ces fonctions.

```
Nom                 : JOYS
Adresse du point d'entrée : $E827
Paramètre d'entrée  : Accu A du 6809 E
Paramètre de retour : Accu B et BIT C du RE 6809 E
Effet               : Effectue une lecture de la manette
                      désignée dans A et retourne sa
                      position dans B
Exemple             : Après avoir lancé le programme ci-dessous, inclinez
                      la manette sur l'une des huit positions et appuyez
                      sur le bouton ACTION.

Le bouton ACTION entraîne le SWI. Par la commande R du moniteur (cartouche
ASSEMBLEUR TOTEK), vérifiez le code dans B.

        ORG   $A000
JOYS    EQU   $E827
        CLRA
ENCOR   ISR   JOYS
        BCC   ENCOR
        SWI
        END
```


8. Gestion de l'interface de communication

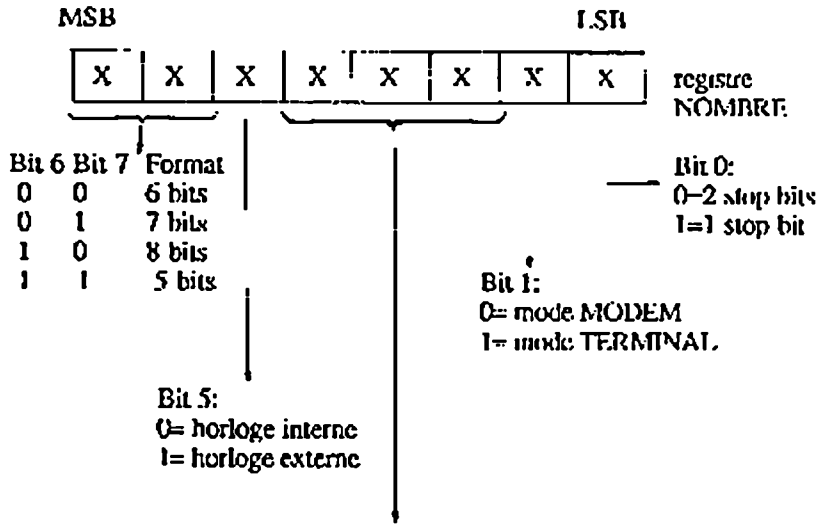
La routine RSCO permet d'envoyer (ou de recevoir) des valeurs transférées en série ou en parallèle, sur l'interface de communication. Les registres à paramétrer sont nombreux, leur contenu précise le mode d'échange choisi, l'état de la communication ou la vitesse de transmission.

Le registre RS.OPC ($602B) sélectionne le mode de transfert désiré, les codes sont à choisir parmi les suivants:

Opération désirée	Valeur à implanter dans RS.OPC
Ouverture en lecture-écriture RS232	$01
Lecture d'un caractère en RS232	$02
Ouverture en écriture seule	$04
Ecriture d'un caractère en série	$09
Ecriture d'un caractère en série	$0C
Fermeture en série	$11
Fermeture en série	$14
Ouverture en écriture parallèle	$40
Ecriture d'un caractère en parallèle	$08
Fermeture en parallèle	$10
Copie graphique d'écran	$20

En cas d'écriture, l'accu B doit contenir l'octet à envoyer; en cas de lecture l'accu B contient le caractère reçu. Si seule la ligne série est ouverte, une fermeture en parallèle sera considérée comme une fermeture série. Pour la copie graphique de l'écran, le registre GRCODE ($6047) doit contenir le code de mise en mode graphique spécifique à l'imprimante utilisée. GRCODE contient la valeur 7 par défaut. La copie graphique s'exécute selon le mode 40 ou 80 colonnes.

Pour les transmissions série, vous préciserez le format des données échangées, en programmant le registre NOMBRE (S6046) selon les options suivantes:



Bit 6	Bit 7	Format
0	0	6 bits
0	1	7 bits
1	0	8 bits
1	1	5 bits

Bit 4	Bit 3	Bit 2	Choix correspondant
0	0	0	Pas de parité
0	0	1	Parité impaire
0	1	1	Parité paire
1	0	1	Mark bit
1	1	1	Space bit

Enfin, vous choisirez la vitesse de transmission en programmant le registre BAUDS (S6044-$6045).

Vitesse désirée:	Valeur à implanter dans BAUDS
50 bauds	$0001
75 bauds	$0002
110 bauds	$0003
135 bauds	$0004
150 bauds	$0005
300 bauds	$0006
600 bauds	$00CA
1 200 bauds	$0008
1 800 bauds	$0009
2 400 bauds	$000A
3 600 bauds	$000B
4 800 bauds	$000C
7 200 bauds	$000D
9 600 bauds	$000E
19 200 bauds	$000F

Cependant, pour la compatibilité avec le TO7, ce paramètre peut être pris (pour certaines vitesses) dans la table BDTAB située à l'adresse $E836. Le premier paramètre (sur 2 octets) représente la vitesse pour 110 bauds, les suivants pour 300, 600, 1 200, 2 400 et 4 800 bauds.

Le registre RS.STA en retour de la routine RSCO nous indique l'état de la communication. L'interprétation de son contenu est conforme au tableau ci-dessous:

Contenu de RS.STA	Interprétation de l'état de la communication
$01	Ouvert en lecture-écriture RS232
$04	Ouvert en écriture seule RS232
$10	Fermé
$40	Ouvert en écriture parallèle
$80	Périphérique non prêt

Le bit de retenue (C) du registre d'état 6809 E est forcé à 0 si tout s'est passé normalement, et à 1 dans le cas contraire.

Nom	: RSCO
Adresse du point d'entrée	: $E812
Paramètres d'entrée	: Accu B du 6809 E (en cas d'écriture) Registre RS.OPC $602B Registre BAUDS $6044-$6045 Registre NOMBRE $6046 Registre GRCODE $6047 (TO9)
Paramètres de retour	: Accu B du 6809 E (en cas de lecture) BIT C du RE 6809 E Registre RS.STA $602C
Effet	: Permet la lecture ou l'écriture en série RS232, ou l'écriture en mode parallèle.

9. Gestion du lecteur-enregistreur de cassettes

La routine K7CO permet de gérer les échanges de données entre l'unité centrale et le lecteur-enregistreur de cassettes. A cet effet, le registre K7.OPC (S6029) permet de spécifier, par le choix de son contenu, l'opération désirée:

Opération désirée	Valeur à implanter dans K7.OPC
Ouverture en lecture	$01
Lecture d'un caractère	$02
Ouverture en écriture	$04
Ecriture d'un caractère	$08
Fermeture	$10

Dans le cas d'une écriture sur la cassette, l'accu B doit contenir l'octet à envoyer avant l'appel de la routine. Dans le cas d'une lecture, l'accu B reçoit le caractère. En retour de la routine K7CO, le registre K7.STA contiendra le code de l'opération réalisée:

Contenu de K7.STA	Interprétation
$01	Ouvert en lecture
$04	Ouvert en écriture
$10	Fermé
$80	Périphérique non prêt

En cas d'erreur ou si, par exemple, le lecteur n'est pas connecté à l'unité centrale, la routine K7CO positionne automatiquement le BIT C du registre d'état microprocesseur à 1.

Nom	: K7CO
Adresse du point d'entrée	: $E815
Paramètre d'entrée	: Accu B du 6809 E (en cas d'écriture) Registre K7.OPC $6029
Paramètre de retour	: Accu B du 6809 E (en cas de lecture) BIT C du RE 6809E Registre K7.STA $602A
Effet	: Permet d'écrire ou de lire une donnée sur le périphérique cassette.

10. Contrôleur de disquettes

Les disquettes 3,5 pouces utilisées par les TO8, TO9 et TO9+ sont divisées en 80 pistes de 16 secteurs chacune. Au gré de l'utilisateur, elles peuvent être formatées en simple ou double densité ce qui modifiera le format des secteurs:

simple densité = 128 octets par secteur
double densité = 256 octets par secteur.

En conséquence, le contrôleur de disquettes intégré aux unités centrales peut indifféremment travailler dans les deux modes. Nous pouvons gérer les échanges de données entre le microprocesseuer et le lecteur de disquettes selon deux principes:

– au niveau physique, en utilisant le point d'entrée du moniteur
– au niveau logique, en manipulant des fichiers au format BASIC Microsoft.

Nous vous proposons d'étudier ces deux possibilités.

Gestion physique

Les routines DKFORM et DKCO permettent respectivement de formater une disquette et de lire ou écrire des données. Le nombre de registres constituant les paramètres d'entrées ne doivent effrayer personne; globalement leurs rôles sont les suivants:

DK.OPC contient le code de l'opération demandée
DK.DRV repérage du numéro de lecteur concerné
DK.SEC repérage du numéro de secteur concerné
DK.TRK repérage du numéro de piste concerné
DK.BUF repérage de la zone tampon d'écriture-lecture

Donc, par la programmation du registre DK.OPC vous définirez la tâche à réaliser. Les codes seront choisis parmi les suivants:

Travail à réaliser	Code à implanter dans DK.OPC
Formatage (sans vérification)	$00
Initialisation du contrôleur	$01
Lecture d'un secteur	$02
Positionnement en simple densité	$04
Ecriture d'un secteur	$08
Positionnement en double densité	$10

Recherche de la piste 0	$20
Recherche de piste (repérée par DK.TRK)	$40
Vérification en écriture	$80

Remarques:
– Si l'initialisation s'est déroulée sans problèmes, le bit de retenue (ou Carry C) du registre d'état du microprocesseur est mis à 0 et le type de contrôleur "D" est retourné dans le registre DK.STA. Dans le cas contraire, le bit de retenue est forcé à 1 et le code d'erreur $40 est mis dans le registre DK.STA.
– Pour la vérification en écriture, il est nécessaire de faire un "OU" logique entre le code $80 et le code de l'opération à vérifier.

En fonction du code implanté dans DK.OPC, certains registres ci-dessous devront être initialisés:

• Registre DK.DRV ($6049). Ce registre doit contenir le numéro de lecteur concerné (soit une valeur comprise entre 0 et 4).

Précisons à ce sujet que le lecteur intégré au TO9 ne fonctionne que sur une seule face ainsi que le lecteur stand alone référencé DD09-350. Le lecteur intégré au TO9+ fonctionne sur les deux faces. Le lecteur stand alone référencé DD90-352 raccordé au TO8 par le connecteur DIN 14 broches fonctionne également sur les deux faces. Pour le registre DK.DRV, les numéros 0 et 1 correspondent aux deux faces du lecteur interne, les numéros 2 et 3 aux deux faces du disque externe (donc le 1 et le 3 sont inutilisables sur le TO9). Le numéro 4 est le "RAM disk" ou "disque virtuel" physiquement représenté par l'extension mémoire 64 K pour le TO9 ou de RAM résidente pour les TO8 et TO9+. Pour le TO9, cette interface sera gérée comme une unité de disque double densité, ayant 16 pistes (de 16 à 32) de 16 secteurs contenant 256 octets.

• Registre DK.TRK ($604A $604B). Ce registre doit contenir le numéro de piste où l'on désire travailler; la valeur est comprise entre 0 et 79.

• Registre DK.SEC ($604C). Ce registre doit contenir le numéro de secteur concerné par la lecture ou l'écriture: la valeur est comprise entre 1 et 16.

• Registre DK.NUM ($604D). Ce registre contient l'entrelacement des secteurs logiques sur la disquette lors du formatage. Cette technique permet d'accélérer les temps d'accès au disque. Sur le TO9, l'entrelacement utilisé par le BASIC est de 7. Une lecture rapide des secteurs peut être optimale avec un entrelacement de 2.

• Registre DK.BUF ($604E-$6050). Ce registre contient l'adresse d'origine d'une zone tampon en RAM, d'une capacité de 128 octets en simple densité ou 256 octets en double densité. Ce buffer contiendra soit les données à écrire sur la disquette, soit les données lues sur la disquette.

En retour des routines DKFORM et DKCO , le registre DK.STA peut être interrogé afin de savoir si l'opération s'est bien déroulée. Dans la négative, DK.STA contiendra un code précisant l'erreur détectée (et bit C = 1). Les interprétations sont les suivantes:

Code lu dans DK.STA	Interprétation
$01	Disquette protégée.
$02	Problèmes de timing ou données perdues.
$04	Erreur de secteur, identificateur incorrect. Secteur ne pouvant être lu ou erreur sur le checksum, cependant la piste peut être correcte.
$08	Erreur sur les données, l'identificateur de secteur est correct, mais les données ne peuvent être lues, ou le checksum est incorrect.
$10	Lecteur non prêt; le moteur n'est pas en route ou le lecteur spécifié est inexistant.
$20	Erreur sur vérification. La zone tampon en mémoire et la zone correspondante écrite sur la disquette ne sont pas identiques.

Nom	: DKFORM
Adresse du point d'entrée	: $E007
Nom	: DKCO
Adresse du point d'entrée	: $E82A
Paramètre d'entrée	: Registre DK.OPC $6048 Registre DK.DRV $6049 Registre DK.SEC $604C Registre DK.TRK $604A-$604B Registre DK.BUF $604E-$6050
Paramètre de retour	: BIT C du RE 6809E Registre DK.STA $604E
Effet	: Permet de formater une disquette et d'accomplir des actions de lecture-ecriture.

Format BASIC Microsoft

Les disquettes sont structurées selon le format BASIC Microsoft. Cette règle de base nous assure l'interactivité de fichiers créés sous le contrôle de logiciels différents. Ce format impose que la piste 20 soit une zone réservée et organisée de la manière suivante:

Secteur 1 :	Nom de la disquette sur les 8 premiers octets
Secteur 2 :	Table d'allocation des fichiers (FAT)
Secteur 3 à 16 :	Catalogue

Table d'allocation des fichiers

Les fichiers sont organisés en blok de 1 K en simple densité ou 2 K en double densité. Dans tous les cas nous aurons deux blocs par piste. Les blocs sont numérotés à partir de 0. Chaque octet de la table d'allocation des fichiers, à partir de l'octet 1, représente un bloc physique.

Organisation de la "FAT":

Octet 0 :	0
Octet 1 :	Bloc 0, piste 0, secteurs 1 à 8
Octet 2 :	Bloc 1, piste 0, secteurs 9 à 16
Octet 3 :	Bloc 2, piste 1, secteurs 1 à 8
Octet 4 :	Bloc 3, piste 1, secteurs 9 à 16
.....	.....
Octet 2j-1 :	Bloc 2j-2, piste j-1, secteurs 1 à 8
Octet 2j :	Bloc 2j-1, piste J-1, secteurs 9 à 16
.....	.....
Octet 160 :	Bloc 159, piste 79, secteurs 9 à 16

En simple densité, la FAT est limitée à 127 blocs. Un octet de la "FAT" représentant un bloc physique peut avoir comme valeur:

- $FF, qui signifie bloc non alloué
- $FB, qui signifie bloc reservé
- Tout nombre de 0 a $BF signifie bloc alloué. Dans ce cas, le nombre représente le numéro du bloc logique suivant du même fichier.
- Tout nombre de $C1 à $C8 signifie dernier bloc d'un fichier. Les quatre bits de poids faible indiquent le nombre de secteurs utilisés dans ce dernier bloc.

Le catalogue

Le rôle du catalogue est de donner la liste de tous les fichiers enregistrés sur la disquette. Il occupe à ces fins 14 secteurs. Chaque fichier est répertorié sur 32 octets. Il y a 4 fichiers répertoriés par secteur en simple densité et 8 fichiers par secteur en double densité. En conséquence, le catalogue peut donc répertorier au maximum 56 fichiers en simple densité et 112 en double densité.

Chaque fichier dans le catalogue est memorisé de la manière suivante:

Octets 00 à 07	: Nom du fichier, cadré à gauche et completé par des blancs
Octets 08 à 0A	: Suffixe du fichier (.BAS, .BIN, etc.), cadré à gauche et complété par des blancs.
Octet 0B	: Type de fichier: 0 pour un programme BASIC, ASCII ou binaire 1 pour des données BASIC en ASCII 2 pour un programme en langage machine 3 pour un fichier assembleur édité en ASCII
Octet 0C	: Sémaphore: $FF pour de l'ASCII $00 pour du binaire
Octet 0D	: Numéro du premier bloc logique du fichier
Octets 0E à 0F	: Nombre d'octets utilisés dans le dernier secteur du fichier
Octets 10 à 17	: Commentaire associé au fichier
Octets 18 à 1F	: Réservés.

Le premier octet de chaque entrée dans le catalogue indique son état:

$00	: Entrée non allouée, pas de fichier répertorié pour cette entrée
$20 à $7F	: Code ASCII du premier caractère du nom de fichier, donc entrée allouée
$FF	: Fin logique du catalogue

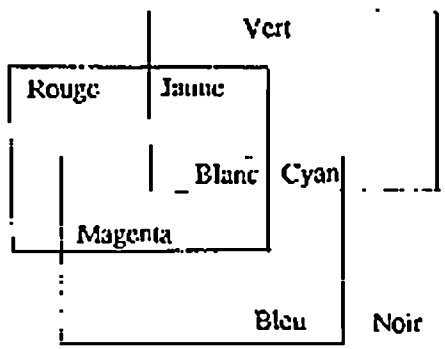
Lors de la création du catalogue, ces octets sont tous mis à $FF. Chaque fois qu'un fichier est créé, la fin logique du catalogue est déplacée dans le premier octet de l'entrée suivante, jusqu'à concurrence de la capacité maximum (catalogue plein). La destruction d'un fichier entraîne la mise à zéro du premier octet de son entrée (l'entrée devient non allouée). Dans ce cas, tout fichier créé ultérieurement se verra attribuer en priorité cette entrée.

11. Programmation de la Palette

La génération d'une couleur est obtenue en dosant judicieusement les teintes fondamentales utilisées en télévision:

- le Rouge
- le Vert
- le Bleu.

Par association (ou dissociation) de ces trois composantes primaires, nous obtenons quatre autres couleurs:



Le noir constituant une 8e couleur est en fait obtenu par l'absence des trois fondamentales.

En jouant sur des intensités plus ou moins importantes de ces trois teintes fondamentales, nous pouvons créer des nuances intermédiaires et obtenir ainsi, par exemple, un jaune plus ou moins foncé.

A chaque couleurs correspond un registre interne du circuit Palette. Nous disposons sur les TO8, TO9 et TO9+ de 16 couleurs utilisables simultanément; en conséquence, Palette possède 16 registres internes différents. Le premier s'appelle 0, son contenu permet de définir la teinte visualisée lors de l'appel de l'attribut de couleur 0, et ainsi de suite jusqu'au registre 15.

Chaque registre est cadré sur 16 bits; le principe du codage est le suivant:

```
MSB                         LSB
  XXXM BBBB VVVV RRRR
```

Les trois bits de poids fort sont indifférents. Le bit M (masque) détermine si la couleur correspondante au registre sera transparente ou non lors d'une incrustation vidéo.

Si M = 0, couleur non transparente.
Si M = 1, cas contraire.

Les trois mots de quatre bits BBBB, VVVV, RRRR permettent respectivement de définir l'intensité de bleu, de vert et de rouge. Il faut, bien sûr, partir du principe que mettre la valeur 0000 dans BBBB aura pour effet de n'avoir pas de bleu dans la teinte. Par contre la valeur 1111 dans BBBB correspondra à un bleu saturé (maximum d'intensité).

Par fondamentale, nous avons donc 2^4 = 16 combinaisons possibles. Ayant trois teintes fondamentales codées sur le même principe, cela fait:
16 × 16 × 16 = 4 096 teintes différentes.

Ecriture-lecture d'un registre couleur

La routine SETP permet d'effectuer toutes les opérations de lecture et d'écriture des registres couleurs du circuit Palette. Le numéro de registre (ou de couleur) est, dans ce cas, implanté dans l'accumulateur A. La technique de programmation dite "des masques" est automatiquement utilisée par la routine SETP. A cet effet le registre X est un masque ET, et le registre Y un masque OU. Nous rapellons que, par principe, le masque ET permet de forcer des 0.
Exemple:

```
  X X X M   B B B B   V V V V   R R R R   Contenu du registre
. 1 1 1 1   1 1 1 1   0 0 0 0   1 1 1 0   Contenu de X
-------------------------------------------------------------
  X X X M   B B B B   0 0 0 0   R R R 0   Résultat
```

Alors que le masque OU permet de forcer des 1.
Exemple:

```
  X X X M   B B B B   V V V V   R R R R   Contenu du registre
+ 0 0 0 0   0 0 0 0   1 1 1 1   0 0 0 1   Contenu de Y
-------------------------------------------------------------
  X X X M   B B B B   1 1 1 1   R R R 1   Résultat
```

Donc par association de X et de Y, vous pouvez redéfinir chaque bit de chaque registre couleur. En fait, l'opération logique réalisée par SETP est:

REGISTRE = (REGISTRE . X) + Y

Prenons un exemple: on désire changer le contenu du registre 0 et obtenir à la place du noir, un jaune demi teinte. Il faut alors mélanger du rouge demi-teinte avec du vert demi-teinte.
Soit:

BBBB	= 0000	Pas de bleu
VVVV	= 0111	Moitié de vert
RRRR	= 0111	Moitié de rouge

Les opérations binaires sont:

X X X M	B B B B	V V V V	R R R R	Cont. init. reg. 0
. 1 1 1 1	0 0 0 0	0 1 1 1	0 1 1 1	Contenu de X
X X X M	0 0 0 0	0 V V V	0 R R R	Résult. intermédiaire

Puis

X X X M	0 0 0 0	0 V V V	0 R R R	
+ 0 0 0 0	0 0 0 0	0 1 1 1	0 1 1 1	Contenu de Y
X X X M	0 0 0 0	0 1 1 1	0 1 1 1	Résultat final

Donc pour cette teinte, le registre X doit contenir la valeur $F077, le registre Y la valeur $0077, et l'accumulateur A la valeur 0 (registre numéro 0), cf. le programme PALETTE ci-contre.

Pour lire le contenu d'un registre sans le modifier, X doit contenir $FFFF et Y doit contenir $0000.

```
Nom                    : SETP
Adresse du point d'entrée : $EC00
Paramètre d'entrée     : Accu A
                         Registre X
                         Registre Y du 6809 E
Paramètre de retour    : Registre X
Effet                  : Permet d'écrire dans le registre Palette désigné par A
                         la valeur binaire résultant d'un masquage ET et OU
                         des registres X et Y. Cette opération est effectuée
                         sans attente de retour trame. En cas de lecture, la
                         valeur est à lire dans le registre X.
Exemple                :
```



```
* REMPLACEMENT DU NOIR PAR UN JAUNE
* DEMI TEINTE.

        TITLE   PALETTE
        ORG     $A000
STEP    EQU     $EC00
        CLRA                    COUL=0
        LDX     #$F077
        LDY     #$0077
        JSR     STEP
        SWI
        END
```

Programmation complète de la palette

Si vous voulez redéfinir le contenu de l'ensemble des 16 registres couleurs du circuit Palette, vous pouvez évidemment refaire la procédure décrite ci-dessus16 fois de suite !! Mais il y a plus simple. Le registre A doit contenir dans ce cas la valeur $FF. Le registre X doit pointer une table de 32 octets qui représentent les 16 valeurs à implanter dans les registres. Les deux premiers octets codent la couleur 0, les deux derniers codent la couleur 15 (ceci en mode 40 colonnes).

Vous appelez ensuite une seule fois la routine SETP, et l'ensemble des registres seront reprogrammés par un transfert automatique du contenu de la table dans les registres Palette.

```
Nom                    : SETP
Adresse du point d'entrée : $EC00
Paramètre d'entrée     : Accu A (valeur $FF)
                         Registre X du 6809 E
Paramètre de retour    : Néant
Effet                  : SETP prend des valeurs dans une table pointée par
                         X, et les charge automatiquement dans les 16
                         registres couleurs. Cette opération est effectuée avec
                         attente de retour trame.
Exemple                : Le programme ci-dessous peut se scinder en 2. Le
                         premier, PROG1, implante 32 octets à partir de
                         l'adresse $B000. Le second, PROG2, réalise la
                         programmation complète de la palette (transfert des
                         octets de la table dans les registres couleurs). Notez
                         que les valeurs correspondent aux 16 possibilités de
                         la fondamentale Rouge. Le résultat final sera un
                         dégradé du rouge-noir au rouge saturé que vous
                         pourrez visualiser après un RESET dans le menu
                         Réglages et Préférences.
```

```
        TITLE   PALET2
        ORG     $A000
SETP    EQU     $EC00
PROG1   EQU     *       ECRIT VAL TABLE.
        LDB     #$01
        LDY     #$B000
        LDX     #$0000
ECRIT   STX     ,Y++
        ABX
        CMPY    #$B020
        BNE     ECRIT

PROG2   EQU     *       IMPLANT VAL PALET
        LDA     #$FF
        LDX     #$B000
        JSR     SETP
        SWI
        END
```

Correspondance mode d'affichage-registres couleurs

Précédemment, nous sommes partis du principe que la couleur appelée par l'attribut 0 correspondait au contenu du registre 0 de couleur Palette, et ainsi de suite jusqu'à 15. Mais ce raisonnement n'est exact que pour le mode TO7/70 (40 colonnes, 320 × 200 points, 16 couleurs).

Or, les TO8, TO9 et TO9+ possèdent d'autres modes d'affichage que nous avons déjà détaillé (80 colonnes, Bit-map, etc.), et qui nous obligent à structurer d'une manière différente le traitement vidéo (cf. étude matérielle). Pour une bonne programmation du circuit Palette, la table suivante doit être utilisée afin de connaître, en fonction du mode d'affichage utilisé, les rôles et attributions de chacun des registres couleurs.

Modes	Numéro de couleur															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
40 col.	Ff	Ff	Ff	Ff	Ff	Ff	Ff	Ff	Ff	Ff	Ff	Ff	Ff	Ff	Ff	Ff
80 col.	f	F	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Page 1	f	-	F	-	-	-	-	-	-	-	-	-	-	-	-	-
Page 2	f	F	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Superpos.	f	F2	F1	-	-	-	-	-	-	-	-	-	-	-	-	-
Bitmap 4	F	F	F	F	-	-	-	-	-	-	-	-	-	-	-	-
Bitmap 16	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
Triple sup.	f	F1	F2	-	F3	-	-	-	F4	-	-	-	-	-	-	-

Légende: f = couleur de fond F = couleur de forme = cases inaccesibles

En programmation complète de la palette, les numéros logiques ne correspondent plus aux numéros physiques. Il faut donc accéder à la table de conversion suivante pour modifier les couleurs selon le mode sélectionné.

Modes	Numéro de couleur															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
40 col.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
80 col.	8	14	10	11	12	13	9	15	0	1	2	3	4	5	6	7
Page1	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7
Page2	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7
Superpos.	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7
Bitmap 4	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7
Bitmap 16	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7
Triple sup.	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7

Fichier PALETTE.CFG

Le fichier PALETTE.CFG contient les données d'une palette sauvegardée par l'utilitaire "Choisir sa palette de couleurs" issu du menu de garde "Réglages et Préférences ". Il s'agit d'un fichier binaire de 32 octets correspondant aux 16 mots de 16 bits, à transférer dans les 16 registres couleurs du circuit Palette. Le cadrage est, bien entendu, identique à celui décrit au début de ce chapitre.

```
MSB                    LSB
XXXM BBBB VVVV RRRR
```


12. Génération de sons

Création d'un bip

La création d'un bip sonore est obtenue en envoyant le code BEL. (S07) à la routine PUTC ($E803), par l'accumulateur B du 6809 E.
Exemple:

```
        ORG   $A000
PUTC    EQU   $E803
        LDB   #$07
        JSR   PUTC
        SWI
        END
```

Création musicale

La routine NOTE permet d'écrire des compositions musicales que jouera docilement votre micro-ordinateur. D'une manière générale, chaque note sera caractérisée par:

- son identification
- son octave
- sa durée
- son timbre

et l'ensemble du morceau musical sera conditionné par le tempo.

L'identification de la note à jouer sera précisée dans l'accumulateur B du 6809 E.Vous avez 13 notes à votre disposition, de DO à UT, plus le silence. Le tableau ci-dessous donne la correspondance entre les notes désirées et les codes à envoyer.

Note désirée	Code à envoyer dans B
Silence	$30
DO	$31
DO#	$32
RE	$33
RE#	$34
MI	$35
FA	$36
FA#	$37
SOL	$38
SOL#	$39
LA	$3A
LA#	$3B
SI	$3C
UT	$3D

L'octave permet de situer la hauteur de la note au sein de la gamme générée par l'unité centrale. Cinq octaves sont disponibles, de l'octave 1 la plus grave à l'octave 5 la plus aiguë. Les codes à implanter dans le registre OCTAVE ($6036-$6037) sont à choisir parmis les suivants:

Octave désirée	Code correspondant
1	$0010
2	$0008
3	$0004
4	$0002
5	$0001

La durée de la note sera indiquée dans le registre DUREE ($6033-$6034). Notez que les codes correspondants suivent une progression arithmétique identique à la durée relative des notes. Ainsi, comme une blanche vaut deux noires, le code de la blanche étant 48, celui de la noire est 24.

Durée de la note	Valeur correspondante
Ronde	96
Blanche pointée	72
Blanche	48
Noire pointée	36
Noire	24
Croche pointée	18
Croche	12
Double croche pointée	09
Double croche	06
Triple croche pointée	05
Triple croche	03
Dans un triolet:	
Noire	16
Croche	08
Double croche	04
Triple croche	02

Le coefficient appelé timbre et chargé dans le registre du même nom ($6035) peut varier de 0 à 5. Physiquement, il permet de modifier le rapport cyclique du signal créant le son, modifiant ainsi son taux d'harmoniques. La valeur 0 correspond à une note "plate".

Le tempo détermine la vitesse générale d'exécution du morceau. C'est comme le métronome que l'on règle plus ou moins vite et qui change la base ou référence de temps, sans changer la durée relative des notes (une croche sera toujours une croche). La valeur du tempo est chargée dans le registre TEMPO ($6031-$6032) et peut varier entre 1 et 255.

Nom	: NOTE
Adresse du point d'entrée	: $E8E8
Paramètres d'entrée	: Accu B du 6809 B Registre OCTAVE $6036-$6037 Registre DUREE $6033-$6034 Registre TIMBRE $6035 Registre TEMPO $6031-$6032
Paramètres de retour	: Néant
Effet	: Joue la note écrite dans B, selon les caractéristiques implantées dans les registres d'entrée.

13. Commutation des mémoires ROM

Le TO9 possède cinq boîtiers ROM accessibles entre les adresses 0000 et 3FFF (IW56, 40, 39, 38 et la cartouche extérieure). Ces boîtiers, hormis la cartouche, contiennent l'ensemble des logiciels intégrés au TO9:

- FICHES ET DOSSIERS
- PARAGRAPHE
- BASIC 128
  BASIC 1.0
- REGLAGES ET PREFERENCES
- EXPLOITATION DE FICHIERS

et L'EXTRAMON.

Cet espace adressable de 16 Ko ne convient pas à tous les logiciels. Certains en effet dépassent cette capacité et sont alors organisés en banques commutables de 16 Ko. La routine COMS permet d'accéder à n'importe quel sous-programme implanté dans n'importe quelle banque de n'importe quel boîtier ROM. Dans ce cas, le registre U du microprocesseur 6809 E doit contenir l'adresse du début du programme à exécuter, et l'accumulateur A contient un octet défini de la manière suivante:

00SS00BB

SS repère le numéro de boîtier et BB le numéro de banque. Le tableau ci-dessous vous aidera dans ces choix.

Programme	SS	BB
Cartouche externe	11	XX
Fiches et dossiers	10	XX
Paragraphe	01	XX
BASIC 128	00	00
Extramon	00	01
BASIC 1	00	10
Exploitation fichiers	00	11

Chaque ROM possède à l'adresse $20 son numéro de banque. Ainsi le moniteur peut s'assurer de l'identité de la banque sélecté, afin de mieux se repérer dans les boîtiers constitués de plusieurs banques de 16 Ko en parallèle. Dans le cas des TO8 et TO9+, le tableau suivant doit être utilisé pour la détermination du contenu de A:

Programme	SS	BB
Cartouche externe	11	XX
BASIC 512	00	00
Extramon	00	01
BASIC 1	00	10
Exploitation fichiers	00	11

Nom	: COMS
Adresse du point d'entrée	: $EC03
Paramètre d'entrée	: Accu A Registre U du 6809 E
Paramètre de retour	: Néant
Effet	: Permet d'accéder à n'importe quel programme en ROM

14. Accès à l'extramoniteur

La routine EXTRA permet d'accéder aux routines de l'extramoniteur. En entrée, l'accumulateur B contient le numéro de la routine à exécuter; les registres nécessaires des pages $6100-$62FF devant être positionnés selon la routine appelée. En sortie, l'accumulateur B contient un numéro d'erreur, les registres des pages $6100-$62FF étant modifiés si nécessaire.

Pour de plus amples détails sur l'extramon, reportez-vous à la sixième partie de cet ouvrage traitant de son utilisation.

Nom	: EXTRA
Adresse du point d'entrée	: $EC0C
Paramètre d'entrée	: Accu B
	Registres $6100-$62FF, selon routine
Paramètre de retour	: Accu B
	Registres $6100-$62FF, selon routine
Effet	: Accès aux routines de l'extramon

15. Gestion des interruptions

Nous avons appris, au travers de l'étude matérielle, que certaines interruptions du 6809 E sont utilisées:

- L'interruption IRQ est déclenchée soit pour le dialogue clavier, soit par le timer du 6846 pour faire clignoter le curseur à l'écran toutes les 100 ms, soit encore pour gérer la souris ou les manettes (TO8 et TO9+).
- L'interruption FIRQ est utilisée pour la gestion du light pen.

Par les différents exemples d'utilisation des routines du moniteur donnés dans les chapitres précédents, nous savons que les interruptions logicielles SWI sont utilisées pour arrêter un programme ou "reprendre la main".

Mais ces différentes interruptions sont programmables. A chacune d'elles correspond un registre en RAM contenant l'adresse du programme qui doit la traiter. A la mise sous tension ou après un redémarrage "à chaud", ces registres ont été initialisés avec l'adresse d'un programme du moniteur. En conséquence, vous pouvez dériver ou aiguiller ces interruptions sur des programmes personnels, en ré-initialisant ces registres !

• Aiguillage des IRQ

L'adresse de votre programme de gestion de l'IRQ générée par le timer doit être implantée dans le registre TIMEPT ($6027-$6028) et dans le registre IRQPT ($6021-$6022).
- Le bit 5 du registre STATUS ($6019) doit être forcé à 1.
  Les registres DP et S doivent être conservés.
- Votre programme doit obligatoirement finir par un JMP KBIN ($E830) pour valider l'interruption.

Si l'interruption IRQ est générée par une autre source que le timer, l'adresse du programme sera implantée en TIMEPT.

Le programme de la page suivante donne un exemple d'une telle procédure. Le programme de gestion de l'IRQ est écrit à partir de l'adresse $A000, et le programme de dérivation est écrit en $7000 (attention, lancez le programme en $7000 et non en $A000). Après lancement, la couleur du cadre changera toute les 100 ms sans pour autant "monopoliser" votre machine.

• Aiguillage des FIRQ
Vous devez mettre l'adresse de votre programme en FIRQPT ($6023 $6024).

• Aiguillage des SWI
Pour gérer les SWI, vous devez mettre l'adresse de votre programme en SWI1 (S602F-S6030). SWI2 saute directement en S6800, et SWI3 en $7000.

Mais attention, certaines routines du moniteur sont interruptibles et vos programmes ne doivent pas modifier leurs paramètres. Un JMP MENU (SE82D) fait revenir à la page d'en-tête.

```
* PROGRAMME DE GEST. IRQ ECRIT EN $A000
* PROGRAMME DE DERIVATION EN $7000

        TITLE   GEST2IRQ
        ORG     $A000
PUTC    EQU     $E803
RETOUR  EQU     $E830
PILE    EQU     $B000
        LDB     #$1B
        JSR     PUTC
        LDB     PILE
        CMPB    #$67
        BNE     SUIT
        LDB     #$60
        STB     PILE
SUIT    JSR     PUTC
        INC     PILE
        JMP     RETOUR

        ORG     $7000
STATUS  EQU     $6019
TIMEPT  EQU     $6027
IRQPT   EQU     $6021
        LDA     STATUS
        ORA     #$20
        STA     STATUS
        LDX     #$A000
        STX     TIMEPT
        STX     IRQPT
        LDA     #$60
        STA     PILE
        SWI
        END
```


16. Initialisation

Le jargon communément employé pour expliquer les processus de fonctionnement logiciel d'une machine utilise les termes de "démarrage à chaud" ou "démarrage à froid" pour qualifier un "reset". Pourtant, un reset sera toujours la conséquence d'un passage à 0 de l'entrée RESET d'un microprocesseur. Alors ?

Le démarrage à froid qualifie le reset crée matériellement et automatiquement par le hard de la machine, à la mise sous tension. Le démarrage à chaud est le reset déclenché volontairement par l'utilisateur, à l'aide du poussoir appelé INIT. Un flag en RAM permet au microprocesseur de faire la différence. La distinction entre les deux implique des tâches différentes effectuées par le programme d'initialisation.

• En cas de reset ou démarrage à chaud, ne seront pas modifiés:

- le réglage du crayon optique
- la programmation du circuit Palette
  le code de mise en mode graphique, spécifique à l'imprimante utilisée
- le contenu du disque RAM.

• En cas de reset ou démarrage à froid, ne seront pas modifiés:

- l'initialisation de la page 0
- le formatage du disque RAM disque s'il existe (TO9)
- la programmation de la palette avec couleurs standards.

• Dans les deux cas, il y a les modifications suivantes:

- les registres d'aiguillage d'interruptions sont initialisés avec les adresses des routines moniteur correspondantes;
- les redirections des routines sont réinitialisées aux valeurs du moniteur, sauf celle du crayon optique;
  les registres contenant les adresses des différentes tables (décodage clavier, générateur de caractères standard, générateur de caractères utilisateur) sont réinitialisés de manière à pointer sur les tables standards;
- le clavier est réinitialisé et son buffer de reception est remis à la valeur initiale;
  les interruptions sont fermées et le curseur est éteint;
- la première banque RAM est sélectée;
- les bits du registre CONFIG ($6074) sont positionnés;
  le MODEM de seconde génération ainsi que toutes les extensions dont le champ d'adressage se trouve entre $E7F0 et $E7FF sont réinitialisés;
- tous les autres registres sont remis aux valeurs standard, le plus souvent à 0.

17. Informations complémentaires

Points d'entrées standard du moniteur

Nom	Point d'entrée	Effet
EXTRA	$EC0C	Appel de l'extramoniteur
PEIN	$EC09	Lecture des boutons du périphérique clavier
GEPE	$EC06	Lecture du périphérique clavier
COMS	$EC03	Appel d'un sous-programme en ROM
SETP	$EC00	Programmation de la palette
CHPL	$E833	Ecriture d'un point "caractère"
KBIN	$E830	Sortie du programme d'interruption
MENU	$E82D	Retour au menu initial
DKCO	$E82A	Contrôleur de disque
JOYS	$E827	Lecture des manettes de jeux
GETS	$E824	Lecture de l'écran
GETP	$E821	Lecture de la couleur d'un point
NOTE	$E81E	Génération de musique
LPIN	$E81B	Lecture du bouton du crayon optique
GETL	$E818	Lecture du crayon optique
K7CO	$E815	Lecture-écriture sur la cassette
RSCO	$E812	Gestion de l'interface de communication
PLOT	$E80F	Allumage ou extinction d'un point
DRAW	$E80C	Tracé d'un segment de droite
KTST	$E809	Lecture rapide du clavier
GETC	$E806	Lecture du clavier
PUTC	$E803	Affichage d'un caractère

Registres du moniteur (page 0)

Détails des attributions de mémoire RAM entre les adresses $6000 et $601F:

Adresse	Nom	Traitement
*$6000 $6015	REDIR	Les routines suivantes du moniteur font une indirection en RAM. Si vous voulez reprendre le contrôle lors d'un appel à l'une de ces routines, il suffit de modifier l'adresse de renvoi dans cette table: $6000-$6001: Indirection de GETLP $6002-$6003: Indirection de LPIN $6004-$6005: Indirection de GETP $6006-$6007: Indirection de GACH $6008-$6009: Indirection de PUTC $600A-$600B: Indirection de GETC $600C-$600D: Indirection de DRAW $600E-$600F: Indirection de PLOT $6010-$6011: Indirection de RSCONT $6012-$6013: Indirection de GETP $6014-$6015: Indirection de GETS
*$6016	PLAN	Numéro du plan dans les modes superposition b2 : Numéro de plan en superposition b1-0 : Numéro de plan en triple superposition
*$6016-$6017	SAVPAL	Sauvegarde de la palette 14 en mode 80 colonnes
*$6019	STATUS	Différents sémaphores: b7 : Semi-graphique b6 : Scroll rapide b5 : IRQ timer validée b4 : Graphique sans écriture de couleurs b3 : Forme seule b2 : Curseur visible ou invisible b1 : Transmission par GETC b0 : Traitement des séquences SS2 dans GETC
*$601A	TABPT	Pointeur dans la table des terminateurs de lignes
*$601B	RANG	Ligne logique courante
*$601C	TOPTAB	Pointeur sur le sommet logique de la table des terminateurs de lignes
*$601D	TOPRAN	Première ligne logique de la fenêtre

*$601E	BOTTAB	Pointeur sur la fin logique de la table des terminateurs de lignes
*$601F	BOTRAN	Dernière ligne logique de la fenêtre
*$6020	COLN	Colonne logique courante
*$6021-$6022	IRQPT	Pointeur sur la routine moniteur de traitement des interruptions IRQ
*$6023-$6024	FIRQPT	Pointeur sur la routine de traitement des interruptions rapides FIRQ
*$6025-$6026	COPBUF	Copie de BUFFAT, réservé au système
*$6027-$6028	TIMEPT	Pointeur sur la routine utilisateur de traitement des intérruptions TIMER
*$6029	K7OPC	Code opération du L.E.P
*$602A	K7STA	Code d'état du L.E.P
*$602B	RSOPC	Mot de commande pour la gestion de la communication
*$602C	RSSTA	Etat courant de la liaison communication
*$602D-$602E	USERAF	Pointeur sur le générateur de caractères utilisateur
*$602F-$6030	SWI1	Pointeur sur SWI
*$6031-$6032	TEMPO	Tempo général pour la musique
*$6033-$6034	DUREE	Durée de la note
*$6035	TIMBRE	Attaque de la note
*$6036-$6037	OCTAVE	Octave de la note
*$6038	FORME	Code de la couleur
*$6039	ATRANG	Sémaphore pour la gestion d'écran b7 : sémaphore de scroll b6 à b2 : réservé b1 : largeur simple ou double b0 : hauteur simple ou double
*$603A	ATRSCR	Sémaphore de gestion plein écran b7 : sémaphore de fond plein écran b6 : sémaphore de forme plein écran b5 à b2 : réservé b1 : largeur simple ou double b0 : hauteur simple ou double
*$603B	COLOUR	Couleur courante
*$603C	TELETL	Si = $F, alors mode page
*$603D-$603E	PLOTX	Abscisse du dernier point
*$603F-$6040	PLOTY	Ordonnée du dernier point
*$6041	CHDRAW	Code ASCII du caractère servant à dessiner
*$6042	CURSFL	Sémaphore de mouvement de curseur
*$6043	COPCHR	Sémaphore de recopie caractère
*$6044-$6045	BAUDS	Paramètre de vitesse de la liaison série
*$6046	NOMBRE	Définition des paramètres de la liaison série
*$6047	GRCODE	Mise en mode graphique de l'imprimante
*$6048	DKOPC	Commande du contrôleur de disque
*$6049	DKDRV	Numéro du disque sélecté
*$604A-$604B	DKTRK	Numéro de piste drive

*$604C	DKSEC	Numéro de secteur drive
*$604D	DKNUM	Entrelacement des secteurs au formatage
*$604E	DKSTA	Etat du contrôleur de disquettes
*$604F-$6050	DKBUF	Pointeur de la zone tampon d'I/O disque
*$6051-$6052	TRACK0	Position de la tête du lecteur 0
*$6053-$6054	TRACK1	Position de la tête du lecteur 1
*$6055-$6056	TEMP1	Registre temporaire
*$6057	TEMP2	Registre temporaire
*$6058	ROTAT	Flag de rotation du moteur
*$6059	SEQUCE	Code indiquant dans quelle séquence de gestion d'écran on se trouve
*$605A-$605B	SCRPT	Pointeur courant dans l'écran
*$605C	SAVCOL	Sauvegarde de la couleur courante
*$605D	ASCII	Code du dernier caractère affiché
*$605E	READCLV	Pointeur de lecture du buffer clavier
*$605F	SCRMOD	Flag indiquant le mode d'affichage
*$6060-$6061	STADR	Adresse du premier octet de la fenêtre
*$6062-$6063	ENDDR	Adresse+1 du dernier octet de la fenêtre
*$6064	TCRSAV	Sauvegarde de l'état courant du timer
*$6065-$6066	TCTSAV	Sauvegarde du compte courant du Timer
*$6067	WRITECLV	Pointeur d'écriture dans buffer clavier
*$6068-$6069	SAVATR	Sauvegarde des attributs courants d'écran
*$606A	US1	Sémaphore des séquences "unit séparator"
*$606B	COMPT	Compteur de caractères répétés
*$606C-$606D	TEMP	Registre temporaire pour le transfert de données
*$606E-$606F	SAVEST	Sauvegarde du pointeur de pile
*$6070	ACCENT	Sémaphore de séquence d'accent
*$6071	SS2GET	Minuscule accentuée
*$6072	SS3GET	Idem
*$6073	BUZZ	Sémaphore d'extinction du buzzer
*$6074	CONFIG	Flag de présence de périphériques
*$6075	EFCMPT	Compteur d'effacement du curseur
*$6076-$6077	BLOCZ	2 octets à 0 pour les inits
*$6078	SCROLS	Sémaphore de scroll doux
*$6079-$607A	BUFCLV	Adresse du buffer de réception clavier
*$607B	SIZCLV	Longueur du buffer clavier
*$607C	ACCES	Validation d'une info périphérique clavier
*$607D	PERIPH	Echo des 3 LSB retournés par le clavier lors de l'envoi d'une commande
*$607E	PERIPH1	Assure la chronologie des infos issues du clavier
*$607F	RUNFLG	Sémaphore indiquant que l'option auto a été choisie
*$6080	DKFLG	Sémaphore de présence du contrôleur disque
*$6081-$6085	IDSAUT	Buffer clavier par défaut
*$6086	CURFLG	Page dans laquelle le curseur clignote en mode 80 colonnes
*$6087	TEMP2	Registre temporaire

*$6088-$608A	RESETP	Adresse d'initialisation des nouveaux périphériques
*$608B-$60CC	STACK	Pile système
*$60CD-$60CE	PTCLAV	Pointeur sur la table de décodage du clavier
*$60CF-$60D0	PTGENE	Pointeur sur le générateur de caractères standards
*$60D1	APPLIC	Checksum de l'application en cours
*$60D2	DECALG	Ajustement du crayon optique
*$60D3-$60FD	LPBUFF	Zone tampon de I/O crayon ou souris
*$60FE-$60FF	TSTRST	Sémaphore de démarrage à chaud ou à froid

Sixième partie

L'extramoniteur

1. Généralités

Principes de base

L'extramoniteur est une bibliothèque de logiciels intégrée aux TO8, TO9 et TO9+ dont le but est de compléter les fonctions gérées par le moniteur décrit précédemment. Il met à notre disposition un panel de routines bien utiles permettant de simplifier considérablement l'écriture de programmes en assembleur et d'optimiser leur temps d'exécution. Les domaines abordés sont variés, il y en a pour tous les gouts: graphisme, tortue, mathématiques, musique et gestion de disquettes.

La procédure pour utiliser des programmes de l'extramoniteur est légèrement différente de celle utilisée pour le moniteur:

1) On appelle l'extramoniteur à l'aide du moniteur en faisant un JSR à l'adresse $EC0C.
2) Par le contenu de l'accumulateur B, on précise quel programme doit être exécuté.

Donc d'une manière générale, l'écriture sera souvent:

```
          .
          .
EXTRA     EQU   $EC0C
          LDB   #$CODE
          JSR   EXTRA
          .
          .
```

Le CODE étant le numéro de la routine concernée.

L'utilisation de cette bibliothèque nécessite néanmoins quelques précautions d'emploi strictes et indispensables, que nous avons rassemblées et appelées les 5 commandements (sic!):

Premier commandement
Avant toute chose, initialiser EXTRAMON avec RESETC (à froid) ou RESETW (à chaud).

Second commandement
Ne pas avoir de sous-programme d'interruption dans l'espace $0000-$4000.

Troisième commandement
Si vous utilisez le disque, il faut indiquer les zones tampons grâce à FCBINI.

Quatrième commandement
Il est important de penser à gérer les erreurs déclarées par EXTRAMON. En conséquence, à la sortie de toute routine il est recommandé (surtout pour les programmeurs inexpérimentés) de tester le registre B. Dans le cas où tout s'est déroulé normalement, l'accu B contiendra 0. Dans le cas contraire, il contiendra un code correspondant aux codes d'erreur du BASIC.

Cinquième commandement
Il faut avoir en mémoire que les paramètres d'entrée sont implantés soit dans les registres du 6809 E, soit dans dans la zone RAM $6100-$62FF. Les paramètres de retour sont, eux, tous en RAM. Tous les registres, sauf B, sont préservés.

Initialisation d'extramon

Conformément au premier commandement, le premier appel d'EXTRAMON doit concerner son initialisation. Cette tâche est réalisée par deux routines:

RESETC pour un reset à froid
RESETW pour un reset à chaud.

Ces routines initialiseront la zone mémoire d'EXTRAMON et feront un test non destructif de la RAM. En retour, la variable NBANK contiendra le nombre de banques RAM présentes dans le système. Le registre MODELE désignera le type d'unité centrale selon le codage suivant:

Code dans Modèle	Unité centrale
$01	TO9
$02	TO8
$03	TO9+

La densité des drives sera initialisée, le registre TYPDSK contenant le type de contrôleur présent selon la formulation suivante:

Code dans TYPDSK	Contrôleur reconnu
$00	Contrôleur simple densité 80K
$01	Contrôleur double densité 5"1/4
$02	Contrôleur double densité 3"1/2
$03	Contrôleur QDD 50K

Nom : RESETC
Code d'entrée : 00
Paramètre d'entrée : Néant
Paramètre de retour : Registre NBANK $618C
Registre MODELE $627B
Registre TYPDSK $6219 (TO8, TO9+)

Effet : Initialise "à froid" la zone RAM d'EXTRAMON, retourne dans NBANK le nombre de banques RAM disponibles, dans MODELE le type d'unité centrale et dans TYPDSK le type de contrôleur.

Nom : RESETW
Code d'entrée : 01
Paramètre d'entrée : Néant
Paramètre de retour : Toute la zone RAM concernée par EXTRAMON
Effet : Initialise "à chaud" la zone RAM d'EXTRAMON:
Exemple :

```
        ORG   $A000
EXTRA   EQU   $EC0C
RESETW  EQU   $01
        LDB   #RESETW
        JSR   EXTRA
        SWI
        END
```

2. Le graphique

Généralités

Avant d'appeler une routine graphique, il est impératif:
- d'appeler une fois la routine CHOIX pour choisir son mode de tracé,
- de définir sa fenêtre de travail.

La plupart des routines graphiques travaillent à partir du point de coordonnées XXXX,YYYY appelé curseur graphique et correspondant au centre des ellipses, à la première extrémité des droites, des rectangles, etc.

La fenêtre de travail

L'EXTRAMON ne détermine pas implicitement de fenêtre de travail. Afin d'afficher du graphisme à l'écran, il est donc indispensable de définir ses dimensions. La fenêtre de travail est déclarée en initialisant directement les registres XL,YB et XR,YT (XLeft, YBottom, XRight, YTop). Aucun appel à EXTRAMON n'est donc à effectuer. En conséquence, faites attention car il ne vérifie rien. Ainsi:

XL ($61A5-$61A6) définit la marge gauche.
YB ($61A7-$61A8) définit la marge haute.
XR ($61A9-$61AA) définit la marge droite.
YT ($61AB-$61AC) définit la marge basse.

Pour travailler sur l'écran complet en 40 colonnes, il faut mettre :
XL = 0, XR = 319, YB = 0, YT = 199.

Choix du type de tracé

Trois types d'écrans sont compréhensibles par l'EXTRAMON du TO9, et quatre par l'EXTRAMON des TO8 et TO9+:

Le plan 320 × 200 16 couleurs dit mode TO7-70
Le plan 640 × 200 bicolore dit 80 colonnes
Le plan 320 × 200 4 couleurs dit bit-map 4 couleurs
Le plan 160 × 200 16 couleurs dit bit-map 16 (TO8 et TO9+)

Ces plans doivent être préalablement choisis par une séquence d'échappement ESC de la routine PUTC du moniteur. La routine CHOIX permet alors de déterminer le type de tracé utilisé par les graphismes. Les options sont les suivantes:

Contenu de TRATYP	Tracé correspondant
$00	Tracé normal
$01	Mode transparent (OU logique)
$02	Mode: inversion (OU exclusif)
$03	Mode ET (TO8 et TO9+ uniquement)

La routine CHOIX sera sollicitée pour tout changement de mode. Le type de tracé sera implanté dans le registre TRATYP, et le graphisme prendra la couleur désignée dans le registre COULEUR (-16 à +15). Le drapeau WITH précise si le tracé sera avec la couleur de fond :

$00	avec la couleur de fond
$FF	sans la couleur (forme uniquement).

Dans le mode TO7-70, des couleurs négatives feront tracer en fond au lieu de forme. Si WITH est à $FF, seul le plan mémoire FORME est affecté.

En mode bit-map 4 couleurs, seuls les deux bits de poids faible de COULEUR sont pris en compte. L'octet WITH à $FF implique que l'écriture ne sera réalisée que dans le plan FORME (ici plan ROUGE).

En mode 80 colonnes, seul le signe de COULEUR agira sur le tracé (tracé de 1 ou de 0). L'octet WITH n'a pas d'effet.

Nom	: CHOIX
Code d'entrée	: 28
Paramètres d'entrée	: TRATYP $61A0 COULEUR $61A9 WITH $6288
Paramètre de sortie	: Néant
Effet	: Détermine le type de tracé utilisé par les graphismes.

Tracé d'un point

La routine PSETXY permet d'afficher un point graphique dont les coordonnées sont exprimées dans le registre X (abscisse) et le registre Y (ordonnée) du 6809 E. En retour, les registres XXXX et YYYY contiendront les coordonnées du dernier point tracé. Précisons que le point ne sera tracé que s'il est à l'intérieur de la fenêtre de travail.

Nom : PSETXY
Code d'entrée : 25
Paramètres d'entrée : Registre X du 6809 E
Registre Y du 6809 E
Paramètres de sortie : XXXX $61A1-$61A2
YYYY $61A3-$61A4
Effet : Trace un point graphique repéré par X et Y

Exemple : Le programme suivant affiche un point de coordonnées 100,50.

```
* TRACE D'UN POINT GRAPHIQUE PAR
* EXTRAMON

        TITLE   EXPOINT
        ORG     $A000
EXTRA   EQU     $EC0C
RESETW  EQU     01
PSETXY  EQU     25
XL      EQU     $61A5
XR      EQU     $61A9
YB      EQU     $61A7
YT      EQU     $61AB
XXXX    EQU     $61A1
YYYY    EQU     $61A3

FLAG    EQU     *       RESET GENERAL
        LDB     #RESETW
        JSR     EXTRA   reset extramon
        LDX     #0
        STX     XL      declaration
        STX     YB      de
        LDX     #319    la
        STX     XR      fenetre
        LDX     #199    de
        STX     YT      travail

FLAG1   EQU     *       prog etudie
        LDX     #100    colonne=100
        LDY     #50     ligne  =50
        LDB     #PSETXY appel
        JSR     EXTRA
        SWI
        END
```


Tracé de droites

A l'aide de la routine LINE, vous pourrez tracer des lignes graphiques qui seront affichées si leur position est à l'intérieur de la fenêtre de travail. Le principe est de désigner les coordonnées des deux extrémités de la droite avant d'appeler la routine. L'abscisse et l'ordonnée du premier point seront respectivement implantées dans le registre XXXX et YYYY, le dernier point sera précisé dans le registre X (abscisse) et Y (ordonnée) du microprocesseur. En retour de la routine LINE, les coordonnées du dernier point tracé seront automatiquement recopiées dans les registres XXXX,YYYY.

Nom : LINE
Code d'entrée : 26
Paramètres d'entrée : Registre XXXX $61A1-$61A2
Registre YYYY $61A3-$61A4
Registres X et Y du 6809 E
Paramètres de retour : Registre XXXX
Registre YYYY
Effet : Trace une ligne graphique entre les deux points designés par leur coordonnées.
Exemple :

```
* TRACE DE DEUX LIGNES GRAPHIQUES PAR
* EXTRAMON


        TITLE   EXLINE
        ORG     $A000
EXTRA   EQU     $EC0C
RESETW  EQU     01
LINE    EQU     26
XL      EQU     $61A5
XR      EQU     $61A9
YB      EQU     $61A7
YT      EQU     $61AB
XXXX    EQU     $61A1
YYYY    EQU     $61A3


FLAG    EQU     *       RESET GENERAL
        LDB     #RESETW
        JSR     EXTRA   reset extramon
        LDX     #0
        STX     XL      declaration
        STX     YB      de
```

```
        LDX   #319      la
        STX   XR        fenetre
        LDX   #199      de
        STX   YT        travail
FLAG1   EQU   *         prog etude
        LDX   #100      1 point colon=100
        LDY   #50       ligne=50
        STX   XXXX
        STY   YYYY
        LDX   #200      2 point colon=200
        LDY   #150      ligne=150
        LDB   #LINE
        TSR   EXTRA
        LDX   #300      3 point colon=300
        LDY   #50       ligne=50
        LDB   #LINE
        JSR   EXTRA
        SWI
        END
```

Motif de remplissage

Pour tout remplissage d'une figure pleine, il faut fournir un motif. Un motif est une suite de 8 octets avec les mêmes conventions que les caractères graphiques de PUTC. Ce motif est à définir en RAM et le pointeur correspondant sera implanté dans le registre MACP ($627D-$627E).

Tracé de rectangles

La routine BOX permet de tracer des rectangles en désignant les coordonnées de deux sommets opposés. La procédure d'appel est identique au tracé de droite, il suffit d'appeler BOX à la place de LINE. En conséquence, l'abscisse et l'ordonnée du premier sommet sont implantées respectivement dans XXXX et YYYY, le sommet opposé est repéré dans les registres X et Y du 6809 E. De plus, par le contenu du registre FILFLG vous indiquez si le rectangle est vide ou plein :

FILFLG = $00 rectangle vide
FILFLG = $FF rectangle plein (BOXF en BASIC).

Si FILFLG est à $FF, le remplissage se fera avec un motif préalablement défini par une série de 8 octets pointés par le registre MACP (voir ci-dessus, Motif de remplissage). Le remplissage se fait dans le mode courant d'affichage choisi lors de l'appel à la routine CHOIX (normal, transparent ou inversion).


```
Nom                     : BOX
Code d'entrée           : 27
Paramètres d'entrée     : XXXX $61A1-$61A2 Abscisse 1e sommet
                          YYYY $61A3-$61A4 Ordonnée 1e sommet
                          Registre X du 6809 E Abscisse 2e sommet
                          Registre Y du 6809 E Ordonnée 2e sommet
                          Registre FILFLG $61EF
                          Registre MACP $627D-$627E
Paramètres de retour    : Registre XXXX
                          Registre YYYY
Effet                   : Trace le rectangle entre les deux sommets désignés
                          et éventuellement remplissage avec un motif.
Exemple

* TRACE D'UNE EDITEPLEINE.LE MOTIF DE
* REMPLISSAGE EST IMPLANTE EN $B000

        TITLE   EXBOX
        ORG     $A000
EXTRA   EQU     $E00C
RESETV  EQU     01
BOX     EQU     27
FILFLG  EQU     $61EF
MACP    EQU     $627D
XL      EQU     $61A5
XR      EQU     $61A9
YB      EQU     $61A7
YT      EQU     $61AB
XXXX    EQU     $61A1
YYYY    EQU     $61A3

FLAG    EQU     *       RESET GENERAL
        LDB     #RESETV
        JSR     EXTRA   reset extramon
        LDX     #0
        STX     XL      declaration
        STX     YB      de
        LDX     #319    la
        STX     XR      fenetre
        LDX     #199    de
        STX     YT      travail
```

```
FLAG1   EQU     *       prog etudie
        LDA     #$FF
        STA     FILFLG  box plein
        LDX     #$B007
        STX     MACP    posit. pointeur
        LDX     #$B000  implantation
        CLRA            du
        LDY     #MOTIF  motif
REFAI   LDB     A,Y     en
        STB     ,X+     forme
        INCA            de
        CMPA    #$08    damier
        BNE     REFAI
        LDX     #100    1 point:colonne=10
        LDY     #50     ligne=50
        STX     XXXX
        STY     YYYY
        LDX     #200    2 point:colonne=200
        LDY     #150    ligne=150
        LDB     #BOX
        JSR     EXTRA   trace de la boite
        SWI

MOTIF   FCB     $AA,$55,$AA,$55,$AA,$55
        FCB     $AA,$55

        END
```

Tracé d'ellipse

Le centre de l'ellipse est le curseur graphique dont les coordonnées sont exprimées dans le registre XXXX (abscisse) et YYYY (ordonnée). Les rayons horizontaux et verticaux sont initialisés respectivement dans les registres AXEH et AXEV. En appelant la routine CIRCLE vous aurez alors le tracé d'une ellipse complète.

Mais vous avez également la possibilité de dessiner des "camemberts" (arcs d'ellipses) en mettant la valeur $FF dans le registre CAMFLG. Il faut alors donner en radian les 2 angles de l'arc dans ALPHA1 et ALPHA2 (4 octets chacun). Le registre FILFLG a ici le même sens que dans BOX.

Nom : CIRCLE
Code d'entrée : 24
Paramètres d'entrée : Registre XXXX $61A1-$61A2
YYYY $61A3-$61A4
AXEH $61F1)
AXEV $61F0)
FILFLG $61EF
CAMFLG $61F2
ALPHA1 $61F3-$61F6
ALPHA2 $61F7-$61FA
MACP $627D-$627E
Paramètre de retour : Néant
Effet : Dessine une ellipse vide ou pleine, totale ou partielle (camembert).
Exemple :

```
* TRACE D'UNE ELLIPSE PLEINE.LE MOTIF DE
* REMPLISSAGE EST IMPLANTE EN $B000

        TITLE   ELLIPS
        ORG     $A000
EXTRA   EQU     $ECOC
RESETW  EQU     01
CIRCLE  EQU     24
AXEH    EQU     $61F1
AXEV    EQU     $61F0
CAMFLG  EQU     $61F2
FILFLG  EQU     $61EF
MACP    EQU     $627D
XL      EQU     $61A5
XR      EQU     $61A9
YB      EQU     $61A7
YT      EQU     $61AB
XXXX    EQU     $61A1
YYYY    EQU     $61A3


FLAG    EQU     *       RESET GENERAL
        LDB     #RESETW
        JSR     EXTRA   reset extramon
        LDX     #0
        STX     XL      declaration
        STX     YB      de
        LDX     #319    la
        STX     XR      fenetre
        LDX     #199    de
        STX     YT      travail
```

```
PRAGI   EQU   *          prog etudie
        LDA   #$EF
        STA   FILFLG     ellipse pleine
        CLRA
        STA   CAMFLG     ellipse complete
        LDX   #$B007
        STX   MACP       point. pointeur
        LDX   #$B000     implantation
        CLRA             du
        LDY   #MOTIF     motif
REPAI   LDB   A,Y        en
        STB   ,X         forme
        INCA             de
        CMPA  #$08       damier
        BNE   REPAI
        LDX   #160       centre colonne=160
        LDY   #100       ligne=100
        STX   XXXX
        STY   YYYY
        LDA   #100       axe horizont=100
        STA   AXEH
        LDA   #50        axe vertical=50
        STA   AXEV
        LDB   #CIRCLE
        JSR   EXTRA      trace de l'ellipse
        SWI
MOTIF   FCB   $AA,$55,$AA,$55,$AA,$55,$AA,$55
        FCB   $AA,$55
        END
```

Remplissage d'une zone

Le curseur graphique (XXXX,YYYY) est l'origine du remplissage. Toute la zone de la fenêtre connexe de même couleur que le point (XXXX,YYYY) sera peinte à l'appel de la routine PAINT. La couleur de remplissage sera celle définie par le dernier appel à la routine CHOIX. Le motif de remplissage est pointé par le registre MACP ($627D-$627E) comme dans le cas des routines BOX et CIRCLE.

Il est indispensable, pour réaliser cette opération, de fournir une zone de travail initialisée entre DEBZON (début de zone) et FINZON (fin de zone). Si cette place est insuffisante pour la complexité du dessin, une erreur "OUT OF MEMORY" sera générée.

Nom	: PAINT
Code d'entrée	: 29
Paramètres d'entrée	: XXXX $61A1-$61A2, YYYY $61A3-$61A4 DEBZON $616B-$616C FINZON $616E-$616F MACP $627D-$627E
Paramètre de sortie	: Néant.
Effet	: Permet de peindre une surface avec un motif de son choix.

Le Micro-Interpréteur Graphique MIG

Le micro-interpréteur graphique MIG est un outil vraiment extraordinaire dans la mesure où il intègre dans une même routine différentes fonctions étudiées précédemment. La routine MIG permet en effet de tracer des figures complexes en un minimum d'instructions. Les programmes sont ainsi plus simples à écrire, plus courts en mémoire et plus rapides en exécution.

Après avoir choisi votre type de tracé par la routine CHOIX ainsi que votre fenêtre d'affichage, MIG est capable de tracer des points, des droites, des rectangles et des ellipses (pleins ou vides).

Il y a deux manières d'utiliser cette routine:

- le macro-assembleur
- l'assembleur.

Dans les deux cas, les codes à utiliser sont du type code opération suivi d'opérandes. La plupart des commandes de MIG sont doublées de façon à pouvoir travailler sur des opérandes de 8 ou 16 bits, ceci afin de gagner de la place et du temps. Les commandes 16 bits commencent par L (pour Long). Quel que soit votre mode de traitement, voici la liste des macros graphiques de MIG :

REL	Passe en relatif
ABS	Passe en absolu (ces 2 commandes permettent de passer des coordonnées de manière relative ou absolue, au choix...)
LFCUR x,y	Positionne le curseur graphique en x,y. x,y sont ici sur 2 octets chacun.
FCUR x,y	Identique à LFCUR, mais x,y sont sur un octet chacun seulement. Utile pour les petits déplacements.

LPOIN x,y	Ecrit un point en x,y
POIN x,y	Idem mais x,y sur 8 bits
LLINE x,y	Trace une ligne jusqu'en x,y
LINE x,y	Idem mais 2 × 8 bits
LBOX x,y	Trace un rectangle
BOX x,y	Idem mais 2 x 8 bits
LBOXF x,y	Trace un rectangle plein
BOXF x,y	Idem mais 2 x 8 bits
ROND a,b	Trace une ellipse de rayon horizontal a et de rayon vertical b. Les 2 rayons sont des octets.
RONDF a,b	Idem mais l'ellipse est pleine.
MOTIF ad	Fournit un nouveau motif de remplissage: ad pointe sur les 8 octets décrivant le motif.
RMOTIF ad	Idem mais permet d'adresser le motif de manière relative au compteur de programme de MIG. Ceci permet de générer des suites de commandes relogeables.
LRMOTIF ad	Idem à RMOTIF mais en relatif 16 bits.

En plus des commandes graphiques, vous disposerez de quelques instructions de contrôle.

CALL ad	Permet l'appel à un sous-programme. Ce sous-programme est bien entendu écrit lui aussi en MIG et doit se terminer par STOP qui est un retour de sous-programme.
RCALL ad	Idem mais relatif au compteur de programme de MIG
LRCALL ad	Idem mais relatif 16 bits.
DO n	Permet de répéter n fois la séquence qui suit jusqu'au LOOP associé.
MULTIP n	Permet de multiplier l'appel à une fonction. Si vous devez appeler 18 fois de suite LINEC, il vaut mieux appeler: MULTIP 18 LINEC x1,y1, x2,y2,..., x18,y18 L'économie ici réalisée est de 15 octets...

Dans le cas où vous travaillez avec un macro-assembleur, le tableau ci-après donne le listing des codes à implanter en FCB pour créer les macro-instructions.

Instructions de l'interpréteur graphique M.I.G.

```
STOP    MACRO              Retour sous-programme
        FCB     0          au principal
        ENDM
CALL    MACRO   ARG        Appel de sous programme
        FCB     1
        FDB     ARG
        ENDM
RCALL   MACRO   ARG        BSR
        BRA     ARG
        ORG     *-2
        FCB     19
        FCB     ARG-*-1
LRCALL  MACRO   ARG        LBSR
        FCB     20
        FDB     ARG-*-2
        ENDM
DO      MACRO   ARG        Structure de contrôle
        FCB     2,ARG
        ENDM
LOOP    MACRO              Le complément du DO
        FCB     0
        ENDM
MULTIP  MACRO   ARG        Cette macro est généralement
        FCB     3,ARG      suivie d'une suite de FCB
        ENDM
MOTIF   MACRO   ARG        Pointeur vers un pattern
        FCB     4
        FDB     8+(ARG)
        ENDM
RMOTIF  MACRO   ARG
        BRA     ARG+8
        ORG     *-2
        FCB     21
        FCB     8+(ARG)-*-1
        ENDM
LRMOTIF MACRO   ARG
        FCB     22
        FDB     8+(ARG)-*-2
        ENDM
ABSOLU  MACRO              Passe en absolu
        FCB     5
        ENDM
```

```
RELATIF  MACRO                    Passe en relatif
         FCB     6
         ENDM
LFCUR    MACRO   ARGX,ARGY        Positionne le curseur
         FCB     7
         FDB     ARGX,ARGY
         ENDM
FCUR     MACRO   ARGX,ARGY
         FCB     8,ARGX,ARGY
         ENDM
LPOIN    MACRO   ARGX,ARGY        Ecrit un point
         FCB     9
         FDB     ARGX,ARGY
         ENDM
POIN     MACRO   ARGX,ARGY
         FCB     10,ARGX,ARGY
         ENDM
LLINE    MACRO   ARGX,ARGY        Trace une ligne
         FCB     11
         FDB     ARGX,ARGY
         ENDM
LINE     MACRO   ARGX,ARGY
         FCB     12,ARGX,ARGY
         ENDM
LBOX     MACRO   ARGX,ARGY        Trace un rectangle
         FCB     13
         FDB     ARGX,ARGY
         ENDM
BOX      MACRO   ARGX,ARGY
         FCB     14,ARGX,ARGY
         ENDM
LBOXF    MACRO   ARGX,ARGY        Trace un rectangle plein
         FCB     15
         FDB     ARGX,ARGY
         ENDM
BOXF     MACRO   ARGX,ARGY
         FCB     16,ARGX,ARGY
         ENDM
ROND     MACRO   ARGX,ARGY        Trace une ellipse
         FCB     17,ARGX,ARGY
         ENDM
RONDF    MACRO   ARGX,ARGY        Une ellipse pleine
         FCB     18,ARGX,ARGY
         ENDM
```


Pour les moins fortunés (comme nous) qui ne possèdent pas de macro-assembleur, pas de panique! Le MIG peut également être utilisé. Dans le listing ci-dessus vous voyez qu'à chaque instruction de MIG correspond un code en FCB:
– pour ROND c'est 17
– pour STOP c'est 0
– pour BOXF c'est 16 etc.
Bien! Dans le tableau précédent (liste des macros graphiques de MIG) sont indiquées les syntaxes de chaque instruction (nombre d'octets et cadrage). En assembleur, il suffira de rentrer une figure graphique constituée de séquences écrites en FCB pour travailler avec MIG.

Exemple d'application: On veut tracer une figure complexe constituée:
d'une boîte (remplie) de coordonnées 150,50 et 200,100
– d'un cercle de coordonnées 200,100 et 50,50
– d'un segment de droite de coordonnées 50,50 à 150,150

Nous écrirons donc la séquence d'octet suivante:

– 06 pour travailler en relatif
– 16,200,100 pour afficher une boîte pleine selon coordonnées
· 17,50,50 pour afficher un cercle selon les coordonnées 50,50
– 12,150,150 pour tracer la ligne jusqu'à 150,150
– 0 pour arrêter la séquence MIG (indispensable).

Dans sa totalité le programme peut s'écrire de la manière suivante:

```
* TRACE D'UNE BOITE PLEINE, D'UN CERCLE
* VIDE ET D'UNE DROITE PAR LE M.I.G


         TITLE   EXMIG
         ORG     $A000
EXTRA    EQU     $EC0C
RESETW   EQU     01
MIG      EQU     30
XL       EQU     $61A5
XR       EQU     $61A9
YB       EQU     $61A7
YT       EQU     $61AB
XXXX     EQU     $61A1
YYYY     EQU     $61A3
```

```
FLAG    EQU     *       RESET GENERAL
        LDB     #RESETW
        JSR     EXTRA   reset extramon
        LDX     #0
        STX     XL      declaration
        STX     YB      de
        LDX     #319    la
        STX     XR      fenetre
        LDX     #199    de
        STX     YT      travail

FLAG1   EQU     *       prog etudie
        LDX     #150    positionnement
        STX     XXXX    du
        LDX     #50     curseur
        STX     YYYY    graphique
        LDX     #FIGURE charge graphisme
        LDB     #MIG    appel a MIG
        JSR     EXTRA
        SWI

FIGURE  FCB     05      mode relatif
        FCB     16,200,100 BOXF (200,100)
        FCB     17,50,50   ROND
        FCB     12,150,150 LINE-(150,150)
        FCB     $0      stop

        END
```

Si vous travaillez en macro-assembleur (petits veinards) la différence essentielle se situe sur la partie en flag FIGURE qui s'écrira:

```
FIGURE  REL
        BOXF    200,100
        ROND    50,50
        LINE    150,150
        STOP
```

Les néophytes auront certainement remarqué au passage qu'un programme écrit à l'aide des macro instructions a des allures de BASIC.

Nom	: MIG
Code d'entrée	: 30
Paramètres d'entrée	: Registre XXXX $61A1-$61A2 Registre YYYY $61A3-$61A4 Registre X du 6809 E
Paramètre de sortie	: Néant
Effet	: A partir du curseur graphique de cordonnées XXXX et YYYY, trace une figure graphique définie en FCB et pointée par le registre d'index X.
Exemple	: voir le programme EXMIG ci-avant.

Codage et décodage d'images

Codage

Nom	: CODE		
Code d'entrée	: 69 (pour TO9+ et TO8)		
Paramètres d'entrée	:		
PUTFLG	$6249		$00
X0COD	$61D6,Y0COD	$61D7	Coin supérieur gauche
X1COD	$61D8,Y0COD	$61D9	Coin inférieur droit
DEBZON	$616B-$616D		Début de la zone résultat
WITH	$6288		$00 ==> avec la couleur $FF ==> sans la couleur
PASSCD	$61DB		$00 ==> codage virtuel $FF ==> code, range en mémoire
Paramètres de retour	: LSTBYT ($61DC-$61DE) Le 1er octet libre après le codage.		
Effet	: Permet de coder un dessin. Les coins de l'image sont donnés en coordonnées caractères. La zone mémoire est remplie si PASSCD est à $FF, sinon, seul le registre LYTBYT est mis à jour. Ceci permet de tester si une zone mémoire est suffisante pour coder l'image de l'écran.		

Décodage

Nom	: CODE	
Code d'entrée	: 69 (pour TO9+ et TO8)	
Paramètres d'entrée	:	
PUTFLG	$6249	$FF
DEBZON	$616B-$616D	Adresse début du dessin
X0COD	$61D6,Y0COD $61D7	Coin supérieur gauche
WITH	$6288	$00 ==> avec la couleur
		$FF ==> sans la couleur
Paramètre de sortie	: Néant	
Effet	: Permet de décoder une image. Le troisième octet de DEBZON contient le numéro de banque logique (de 1 à 6). Si le coin supérieur gauche est trop à droite, ou trop bas pour que le dessin soit affiché, rien n'est tracé. Le décodage se fait dans le mode choisi par la routine CHOIX et permet donc le "ou exclusif" ainsi que d'autres subtilités (voir CHOIX).	

3. Les tortues

Généralités

Une tortue est un objet graphique (fil de fer) que l'on peut déplacer, agrandir, déformer à volonté. D'une manière générale, pour parler à une tortue en appelant EXTRAMON, il suffit d'implanter un code dans l'accumulateur B correspondant à la fonction désirée et de fournir un pointeur vers le descripteur de la tortue dans le registre Y du 6809 E.

Ce descripteur a une longueur de 14 + n octets, n dépendant de la complexité de la forme de la tortue. La plupart des actions réalisables avec une tortue se font grâce à un appel d'EXTRAMON. La manipulation des commandes est très proche de celle du BASIC 128.

Initialisation

Première étape, initialiser la tortue. Cette opération est réalisée par la routine INITORTUE. Après exécution de cette routine:
- La zone est initialisée.
- La tortue est au centre de l'écran, invisible, crayon levé.
- Echelle normale, direction et rotation nulles, mode "rapide".
- Elle est en forme de triangle isocèle (un peu comme la tortue LOGO).

Nom	: INITORTUE
Code d'entrée	: 39
Paramètre d'entrée	: Registre Y du 6809E
Paramètre de sortie	: Néant
Effet	: Initialise une tortue. Le registre Y pointant une zone de 27 octets au minimum.
Exemple	: Voir programme EXTORTUE page 261.

La visibilité

La routine SHOW affiche ou cache une tortue désignée par le registre d'index Y du 6809 E. Les tortues sont dessinées en mode "OU exclusif" afin que ces opérations soient rapides. La couleur de la tortue est choisie par la routine CHOIX.

En mode TO7/70, la couleur n'est pas mise en vidéo inverse. Ainsi une tortue peut laisser des traces de couleurs. Vous pourrez éviter cet effet également par la routine CHOIX.

Nom	: SHOW
Code d'entrée	: 33
Paramètres d'entrée	: Registre Y du 6809 E Accumulateur A du 6809 E
Paramètre de sortie	: Néant
Effet	: Si A = $00 la tortue désignée par Y s'éffface Si A = $FF la tortue désignée par Y s'affiche
Exemple	: Voir programme EXTORTUE page 261.

Le déplacement

La routine FWD permet de déplacer la tortue désignée par le registre Y dans la direction courante. Ceci correspond à la commande AVANCE de l'interpréteur LOGO. Le déplacement a lieu dans une fenêtre virtuelle sphérique de - 32768 à 32767 en X comme en Y. La partie visible étant définie par la fenêtre de visualisation. Le pas de déplacement compris entre -256 (recul) et +256 (avance) est à préciser dans le registre d'index X.

Nom	: FWD
Code d'entrée	: 37
Paramètres d'entrée	: Registre Y du 6809 E Registre X du 6809 E
Paramètre de sortie	: Néant
Effet	: Déplace une tortue
Exemple	: Voir programme EXTORTUE page 261.

La direction

Les angles de direction sont fournis en 256ème de cercle au registre d'index X, sachant que le sens positif est celui des aiguilles d'une montre. Les actions FWD agiront dans cette nouvelle direction.

La tortue concernée est désignée par le registre d'index Y. Par l'accumulateur A, on précise si l'angle est absolu (A = 00) ou relatif (A = $FF). En relatif, ceci correspond à l'instruction DROITE de l'interpéteur LOGO. En absolu, cela correspond à l'ordre FCAP (fixe cap).

Nom	: HEAD
Code d'entrée	: 34
Paramètres d'entrée	: Registre Y du 6809 E Registre X du 6809 E Accumulateur A du 6809 E
Paramètre de sortie	: Néant
Effet	: Détermine la direction pour le futur déplacement de la tortue.
Exemple	: Voir programme EXTORTUE page 261.

La rotation

Le sens positif est le sens des aiguilles d'une montre. Les angles sont fournis en 256ème de cercle. La forme de la tortue désignée par le registre Y tourne sur elle-même de X 256ème de cercle sans changer de direction. L'angle est écrit dans le registre X, l'accu A précise s'il est absolu (A = 00) ou relatif (A = $FF).

Nom	: ROT
Code d'entrée	: 35
Paramètres d'entrée	: Registre Y du 6809 E Registre X du 6809 E Accu A du 6809 E
Paramètre de sortie	: Néant
Effet	: La tortue désignée par Y pivote d'un angle écrit dans X.
Exemple	: Voir programme EXTORTUE page 261.

La taille

Nous avons la possibilité de modifier les dimensions de la tortue et de créer ainsi de véritables zooms avant, d'où l'appellation de cette routine. L'agrandissement est limité de manière absolue entre 0 et 255.

Il est conseillé de faire attention sur les agrandissements d'objets. En effet, pour des questions de rapidité EXTRAMON ne teste rien. Si un segment d'un objet dépasse 256 pixels, il peut être mal affiché.

Comme pour les autres routines, le registre d'index Y désigne la tortue concernée, le registre d'index X contient la variable (ici la valeur du zoom), et l'accumulateur A indique si l'agrandissement est absolu (A = 00) ou relatif (A = $FF).

Nom	: ZOOM
Code d'entrée	: 36
Paramètres d'entrée	: Registre Y du 6809 E Registre X du 6809 E Accu A du 6809 E
Paramètre de sortie	: Néant
Effet	: Les dimensions de la tortue désignée par Y sont modifiées selon le rapport écrit dans X et A.

La trace

A chaque tortue est associé un crayon qui peut ainsi laisser une trace de ses déplacements comme le font les pneus boueux d'une voiture sur le bitume (!). Habilement maniée, cette fonction permet de dessiner comme avec LOGO.

La routine TRACE demande à la tortue pointée par le registre d'index Y de baisser son crayon (A = $FF) ou de le relever (A = $00). En LOGO, la différence est faite par les primitives BC (Baisse Crayon) et LC (Lève Crayon). Le tracé se fait dans le mode courant choisi avec la routine CHOIX.

Nom	: TRACE
Code d'entrée	: 31
Paramètres d'entrée	: Registre Y du 6809 E Accu A du 6809 E
Paramètre de sortie	: Néant
Effet	: La tortue pointée par Y laisse une trace de ses déplacements ou non, en fonction de l'accu A.

La vitesse

La routine ANIME permet de moduler la qualité de l'affichage en donnant priorité à la vitesse ou au suivi de l'affichage. Deux modes sont en effet à notre disposition:

– le mode lent
– le mode rapide

La différence essentielle est qu'en mode lent on réaffiche la tortue à chaque modification, alors qu'en mode rapide EXTRAMON ne réaffiche que lors d'un déplacement de la tortue.

En particulier, les actions de ZOOM et de ROT sont différées au prochain mouvement de l'objet. Ceci permet de modifier plusieurs paramètres de la tortue assez rapidement.

On peut aussi, pour des questions de vitesse ou pour déplacer la tortue autrement que par FWD, modifier directement certains octets du descripteur. Les modifications étant faites, il faut appeler MOVE (chapitre suivant) pour les voir à l'écran. De cette manière, en un seul appel à EXTRAMON, nous pouvons changer la taille, la position, la rotation, etc. de la tortue.

Nom	: ANIME
Code d'entrée	: 32
Paramètres d'entrée	: Registre Y du 6809 E Accu A du 6809 E
Paramètre de sortie	: Néant
Effet	: La tortue pointée par Y travaillera soit en mode lent (A = $FF) soit en mode rapide (A = $00).

Le positionnement

Pour modifier la forme d'une tortue, il faut changer la forme standard proposée par EXTRAMON. Sauf si vous êtes amateur de sensations fortes du style "transmutations corporelles" (bonsoir docteur!), il est conseillé de demander à la tortue de se cacher avant de modifier sa forme (un peu comme superman) pour éviter de bizarres effets sur votre écran.

Cette forme se trouve dans le descripteur à partir de TFORME (Y +16). En fait, la forme d'une tortue est une série de commandes consistant à avancer, tourner, lever ou baisser le crayon. Elle est donc définie par une série de doublets ou de triplets.

Exemple:

```
TFORME  FDB   0,15,0      LC AV 15 TD 0 (BC)
        FDB   30,117      AV 30 TD 117
        FDB   15,74       AV 15 TD 74
        FDB   30,74       AV 30 TD 74
        FDB   0,0
```

En général, ce seront des doublets (LONGUEUR, ANGLE), mais notons que si la longueur vaut zéro, alors le doublet qui suit sera effectué crayon levé. Deux zéros terminent la description. L'exemple donné ci-dessus correspond à la forme attribuée par défaut aux tortues.

Voici la liste des paramètres modifiables dans le descripteur:

TX	(en Y + 6)	Abscisse de la tortue	(3 octets)
TY	(en Y + 9)	Ordonnée de la tortue	(3 octets)
TROT	(en Y + 12)	Rotation	(1 octet)
TTAI	(en Y + 13)	Taille	(1 octet)
TDIR	(en Y + 14)	Direction	(1 octet)

TX et TY sont connus au 256ème de pixel, le dernier octet représentant la partie fractionnaire de pixel.

Nom : MOVE
Code d'entrée : 38
Paramètre d'entrée : Registre Y du 6809E
Paramètre de sortie : Néant
Effet : Permet de modifier des paramètres de définition des tortues pointées par Y.

La compilation d'une forme

Nom : CMPTORTUE
Numéro du point d'entrée : 40
Paramètres d'entrée : Registre Y du 6809 E, pointe sur la chaîne à compiler
FACLO ($6151), longueur de la chaîne de caractères
Registre X du 6809 E, pointeur sur une zone de rangement

Paramètres de retour : Registre FACLO $6151, longueur du résultat

La syntaxe est celle utilisée dans l'ordre TURTLE de BASIC 128 ou de BASIC 512. Le résultat de la compilation peut immédiatement être interprété par les commandes tortue.

Exemple d'une tortue en mouvement

```
* DEPLACEMENT D'UNE TORTUE.

        TITLE   EXTORTUR
        ORG     $A000
EXTRA   EQU     $ECOC
RESETW  EQU     01
INITOR  EQU     39
SHOW    EQU     33
FWD     EQU     37
HEAD    EQU     34
ROT     EQU     35
XL      EQU     $61A5
XR      EQU     $61A9
YT      EQU     $61AB
YB      EQU     $61A7

FLAG    EQU     *       RESET GENERAL
        LDB     #RESETW
        JSR     EXTRA   reset extramon
        LDX     #0
        STX     XL      declaration
        STX     YB      de
        LDX     #319    la
        STX     XR      fenetre
        LDX     #199    de
        STX     YT      travail

FLAG1   EQU     *       prog etudie
        LDY     #DTOR   designe tortue
        LDB     #INITOR initialisation
        JSR     EXTRA
        LDA     #$FF
        LDB     #SHOW   montre la tortue
        JSR     EXTRA
REC     JSR     TEMPO   temporisation
```

```
        LDX     #15     angle=15
        LDA     #$FF    angle relatif
        LDB     #ROT    tortue pivote..
        JSR     EXTRA   sur elle meme

        LDX     #15     angle direction=15
        LDA     #$FF    angle relatif
        LDB     #HEAD   tortue change..
        JSR     EXTRA   de direction

        LDX     #15     avance de 15
        LDB     #FWD    deplacement tortue
        JSR     EXTRA

        BRA     REC     on recommence

TEMPO   EQU     *       ajuster la vitesse
        LDX     #$9FFF  par contenu de X
BOUC    LEAX    1,X
        BNE     BOUC
        RTS
DTOR    EQU     *
        END
```


4. Les mathématiques

Généralités

L'EXTRAMON sait même compter! L'ensemble de ses possiblités mathématiques sont exposées dans les chapitres suivants. Passé l'effet de surprise, immanquable lorsque l'on apprend qu'il traite les quatres opérations, on plonge dans une béatitude inénarrable en découvrant que les fonctions trigonométriques ne sont pas épargnées (SIN, COS, TAN, ATN...), ainsi que les logarithmes népériens, exponentiel, racine carrée, conversion de bases, etc. Bref, avis aux amateurs, vous avez dans les mains une petite merveille (non, pas le livre, l'unité centrale!)

Toutes ces fonctions mathématiques seront utilisées en relation de registres appelés accumulateurs FAC et ARG. Globalement vous ferez des opérations unaires et binaires. Dans le premier cas l'argument doit se trouver dans l'accumulateur FAC, un appel à SIN ou COS, par exemple, vous rend le résultat ainsi que dans FAC. Dans le second cas, l'argument gauche doit être dans ARG et l'argument droit dans FAC. Les deux arguments seront de même type et précisés dans VALTYP. Le résultat recherché sera dans FAC. Il est conseillé de faire attention car le type du résultat peut être différent du type des opérandes. Surveillez donc bien le registre VALTYP. Si vous désirez faire des calculs en double précision, il faut nécessairement que VALTYP ($6105) soit égal à 8, et que DBLFLG ($6103) soit à $FF.

Description des accumulateurs

Pour fixer rapidement les idées, nous pouvons dire que FAC est l'accumulateur principal, et ARG le secondaire. ARG sert pour les opérations telles que additions, soustractions, multiplications et divisions.

Les réels courts sont codés (sauf signe) sur 4 octets. Le 1er octet est l'exposant E, les 3 autres la mantisse M.

$$\text{nombre} = 0.\text{Mantisse} \times 2^{\text{Exposant} - 128}$$

Exemple: Le nombre 100 décimal s'écrit:

$0.C80000 \times 2^{7}$

et se code en:

Exposant = 128 + 7 = 135 = %10000111
Mantisse = $C80000 = %11001000 00000000 00000000

Pour ces routines, un exposant nul implique que le nombre est égal à 0. En entrée, le nombre doit être normalisé (bit de poids fort de la mantisse = 1). En sortie, EXTRAMON rend également des nombres normalisés.

- Vous trouverez ci-dessous la liste des composants de FAC :

Pour un réel court ou simple précision:

VALTYP ($6105) : type de l'accu (à 4)
FACEXP ($614E) : exposant binaire (de - 128 + 127)
FACHO ($614F) : mantisse 24 bits, poids forts
FACMO ($6150) : mantisse poids moyens
FACLO ($6151) : mantisse poids faibles
FACSGN ($6156) : signe de l'accumulateur

Pour un réel long, c'est identique sauf:

VALTYP ($6105) : à 8
($6152-$6155) : 4 octets supplémentaires suivent la mantisse

Pour un entier 16 bits:

VALTYP ($6105) : à 2
FACMO, FACLO contiennent l'entier
FACEXP, FACHO, FACSGN sont inutiles.

– Vous trouverez ci-dessous la liste des composants de ARG. Elle est très semblable à FAC:

ARGEXP, ARGHO, ARGMO, ARGLO	($6159-$615C)	pour les réels courts (4)
DARGHO	($615D-$6160)	+4 octets pour les réels longs (8)
ARGSGN	($6161)	signe de l'argument pour réels
ARGMO, ARGLO	($615B-$615C)	pour les entiers

Dans FACSGN et ARGSGN seul le bit 7 est utilisé. Ce bit à 1 représente un nombre négatif.

Echanges mémoire et accumulateur

Les réels courts (4) sont codés sur 5 octets dans l'accumulateur, et sur 4 octets en mémoire. Ceci est possible en codant le signe du nombre dans le bit de poids fort de la mantisse, ce bit étant un 1 pour un réel normalisé. Pour les réels longs, on agit de même, mais avec 4 octets de plus pour la mantisse. Les routines fonctionnent aussi pour les entiers, X pointant alors sur 2 octets seulement.

Pour utiliser les routines de transfert bi-directionnelles entre mémoire et accumulateur:
– il faut implanter le type de la variable dans VALTYP,
– puis appeler une des routines de transfert avec le registre d'index X du 6809 E pointant sur la variable à transférer. Soit:

MOVFM	Code d'entrée 62:	Transfert de FAC vers la mémoire pointée par X.
MOVMF	Code d'entrée 63:	Transfert de la mémoire pointée par X vers l'accumulateur FAC
MOVAF	Code d'entrée 64:	Transfert de l'accumulateur ARG vers l'accumulateur FAC.

Liste des fonctions mathématiques

Le tableau ci-dessous dresse la liste des routines mathématiques disponibles et leur code d'entrée respectif.

Nom	Code d'entrée	Fonctionnalité
SGN	41	Rend le signe de l'accumulateur dans FACMO, FACLO. L'accumulateur "entier" vaut ainsi 0, -1 ou 1
INT	42	Rend la partie entière de l'accumulateur.
ABS	43	Rend la valeur absolue de l'accumulateur.
SQR	44	Rend la racine carrée de l'accumulateur. Le résultat est un nombre réel.
LOG	45	Rend le logarithme néperien de l'accumulateur. Le résultat est un nombre réel.
EXP	46	Rend l'exponentielle de l'accumulateur. Le résultat est un nombre réel.

Nom	Code d'entrée	Fonctionnalité
COS	47	Rend le cosinus de l'accumulateur. L'accumulateur est en radians. Le résultat est un nombre réel.
SIN	48	Rend le sinus de l'accumulateur. L'accumulateur est en radians. Le résultat est un nombre réel.
TAN	49	Rend la tangente de l'accumulateur. L'accumulateur est en radians. Le résultat est un nombre réel.
ATN	68	Rend l'arctangente de l'accumulateur.Le résultat est un nombre réel (TO8 et TO9+ uniquement).
FRCTYP	50	Conversion de type. Paramètres d'entrée: – VALTYP le type courant de l'accumulateur FAC – si registre A = 2 résultat entier 2 – si registre A = 4 résultat réel 4 – si registre A = 8 résultat réel 8
FLXER	51	Rend l'entier tronqué d'un réel. Le résultat est un réel
RND	52	Rend un réel court aléatoire (4)
NEGGO	53	Effectue: FAC= - FAC
ADDGO	54	Effectue: FAC= ARG + FAC
SUBGO	55	Effectue: FAC= ARG - FAC
MULTGO	56	Effectue: FAC= ARG * FAC
DIVGO	57	Effectue: FAC= ARG / FAC. Les arguments sont des réels 4 ou 8 octets
EXPGO	58	Effectue: FAC= ARG ^ FAC. Les arguments sont des réels 4 ou 8
IMODO	59	Effectue: FAC= ARG MOD FAC. Les opérandes doivent être des entiers
IDIVO	60	Effectue: FAC= ARG division entière par FAC. Les opérandes doivent être des entiers
FIN	65	Conversion d'une valeur ASCII en binaire Paramètres d'entrée: – registre Y du 6809 E pointe sur ASCII fini par $00 Paramètres de retour: – FAC contient le nombre – VALTYP le type du nombre Les conventions sont similaires à celles de BASIC. FIN accepte aussi les constantes binaires, octales ou hexadécimales.
PUFOUT	66	Conversion d'une valeur binaire en ASCII décimal. Paramètres d'entrée: FAC ($614E) le nombre à convertir – VALTYP ($6105) le type de celui-ci – Registre Y du 6809 E pointe sur un tampon PUMASK ($617C) Flag de PRINTUSING Le flag PUMASK permet d'utiliser le PRINTUSING comme BASIC. Si ce flag est nul, le format de sortie sera choisi par PUFOUT, sinon les octets DPWID et FLDWID permettent de choisir le nombre de chiffres avant et après le point décimal. DPWID ($617A) nombre de chiffres après le point décimal – FLDWID($617B) idem mais avant le point PUMASK ($617C) bit 0 notation scientifique bit 2 signe après le nombre bit 3 force le "+" pour les positifs bit 5 remplit d'"*" au lieu d'espaces bit 7 USING or not USING Paramètre de retour (pour le TO9): – Le registre Y du 6809 E pointe sur l'ASCII Pour récupérer les codes ASCII, il faut sauter les caractères indésirables (inférieurs ou égaux à $20), ensuite vous trouverez votre nombre se terminant par un zéro binaire. Paramètre de retour (pour les TO8 et TO9+): – Le registre FACMO pointe sur l'ASCII. Le registre FACMO pointe sur l'ASCII dans le tampon que vous avez fourni à travers le registre Y. Le nombre initialement implanté dans FAC est détruit après conversion.

HOFOUT 67

Conversion d'une valeur binaire en ASCII hexadécimal ou octal.

Paramètres d'entrée:
– FACMO ($6150-$6151) le nombre à convertir en entier 2
– Registre Y du 6809 E pointe sur le buffer résultat
– Accu A du 6809 E si $00 la sortie est octale si SFF la sortie est hexa.

5. Le DOS

Pour toutes informations relatives aux disquettes elles mêmes, nous vous prions de vous reporter au chapitre "Contrôleur de disquettes", page 207. Le Disk Opérating System est compatible au format Microsoft. Les chapitres suivants présentent les différentes routines de l'extramon pour travailler sur les disquettes (gestion de fichiers, backup, etc.). Notons qu'en double densité, le DOS n'utilise que 255 octets par secteur (sur 256 de disponibles).

Initialisation du DOS

Après initialisation de l'EXTRAMON par un RESETC, il est également nécessaire d'initialiser les variables d'EXTRAMON liées aux disques (densité, nombre de disques, nombre de fichiers, etc.). L'ensemble de ces opérations est réalisé par la routine FCBINI.

Pour changer la densité d'un drive (si le contrôleur le permet), il faut modifier la table TABDEN débutant à l'adresse $621A. Cette table fait 5 octets, chacun étant associé à un lecteur. Pour être en simple densité, il faut mettre $04 dans l'octet correspondant alors que la valeur $10 permet de commuter en double densité. Cette table est automatiquement initialisée par FCBINI en fonction du contrôleur présent. Avec un contrôleur double densité la table est entièrement mise à $10, ce qui est le cas sur le TO9. Remarque: Il est fortement conseillé de ne pas toucher à la densité du disque virtuel 4, celui-ci est forcément double densité.

Nom	: FCBINI
Code d'entrée	: 02
Paramètres d'entrée	: SECBUF ($6197) Pointe sur un buffer pour un secteur, 256 ou 128 octets selon la densité. FATPTR ($6199) Pointe la place libre pour 5 FATS. Il faut DSBLEN (166) octets par FAT. Registre Y du 6809E Nombre de disques que l'on veut utiliser (5 au maximum). Accu A du 6809E Nombre maximum de fichiers que l'on veut pouvoir ouvrir en même temps. Registre X du 6809E Pointe la zone libre pour FCBLEN (281) × A octets. Réserve la place pour A File Control Block.
Paramètre de sortie	: voir texte
Effet	: Initialisation du DOS

Cache disque

Seul sur les machines TO8 et TO9+,le cache disque est disponible. Il permet,comme sur d'autres machines concurrentes, d'accélérer notablement les accès disques. Initialement prévu pour optimiser les utilisations des lecteurs QDD, le cache disque peut également être appelé pour des accès aux lecteurs double face, double densité.

Aucun cache n'est défini par défaut. Pour en déclarer l'existence ,il est nécessaire de positionner le flag ONOFF ($62B0) à $FF et d'adresser une table de descripteurs de caches par le pointeur BLOCS ($62AE). Chaque descripteur a une longueur de 8 octets divisés de la manière suivante:

KADRESSE	2 octets	Adresse du buffer
KBANQUE	1 octet	Numéro de banque
Reservés	5 octets	

Nous pouvons ainsi déclarer autant de caches que nous voulons. Il suffit d'initialiser les 3 premiers octets de chaque descripteur vers une zone mémoire suffisamment grande pour lire une piste (257*16 en double et 129*16 en simple densité).

Le mot KBANQUE représente un numéro de banque logique compris entre 1 et la valeur écrite dans NBANK ($618C). Extramon reconnait la table des descripteurs de cache lorsqu'un 0 est en première position. D'autre part, il suppose que le disque n'est pas retiré du lecteur. Si vous voulez le prévenir d'un risque de changement, il faut positionner le flag RSKCHG ($6189) à $FF. Extramon considèrera alors ses caches comme incorrects et ira relire le disque si nécessaire. Enfin, EXTRAMON vide ses caches dans les opérations importantes tel que CLOSE, KILL, etc.

Le type de contrôleur est dans l'octet TYPDSK ($6219). Ainsi le programmeur qui préfère la sécurité à la vitesse peut demander un cache sur QDD et pas sur FDD.

Nom	: INICACHE
Code d'entrée	: 70
Paramètre d'entrée	: Flag ONOFF $62B0. Registre BLOCS $62AE Registre RSKCHG $6189
Paramètre de retour	: Néant.
Effet	: Positionne un cache disque après avoir mis à jour les pointeurs et ONOFF=$FF. Si des opérations sont à réaliser sans cache, il suffit de mettre ONOFF à 0. Par exemple, on peut ainsi interdire les caches en écriture et les autoriser en lecture.

Ouverture d'un fichier

Nom	: OPEN	
Code d'entrée	: 03	
On peut ouvrir un fichier selon trois modes: – Lecture, – Ecriture, – Accès direct en lecture/écriture.		
Paramètres d'entrée	: DK.DRV ($6049)	Numéro de drive
	FILMOD ($624B)	Type d'accès M.SQI ($10) ouvre en input. M.SQO ($20) ouvre en output. M.RND ($40) ouvre en direct.
	FILNAM ($624F-$6259)	Nom du fichier, 11 caractères.
	OPTBUF ($625A-$6261)	Commentaire (écriture seule), 8 octets, s'arrête au 1er $00.
En écriture, 2 autres flags:		
	FILTYP ($624C)	type de fichier.
	ASCFLG ($624D)	$FF fichier ASCII. $00 fichier binaire.
Pour l'ouverture d'un fichier en accès direct, nous vous prions de vous reporter à l'étude de PUTGET (page 272).		
Paramètres de retour	: FCBNUM ($6244)	Numéro logique de fichier.
En lecture, 2 autres flags:		
	FILTYP ($624C)	Type de fichier.
	ASCFLG ($624D)	$FF fichier ASCII. $00 fichier binaire.
Remarque	: Un numéro de fichier est rendu dans FCBNUM, il permet de distinguer les différents fichiers ouverts en même temps. Pour lire, écrire ou fermer un fichier, il suffit de mettre son numéro dans FCBNUM avant l'appel à la routine concernée. Tout ceci est transparent si l'on n'ouvre qu'un seul fichier.	

Lecture d'un caractère

Nom	: INPUT	
Code d'entrée	: 05	
Paramètres d'entrée	: FCBNUM ($6244)	Numéro logique de fichier
Paramètres de retour	: CARCOU ($6196)	Le caractère lu
	EOFFLG ($6178)	Flag de fin de fichier
Remarque	: Si EOFFLG est mis, il n'y a pas de caractère valide dans CARCOU.	

Ecriture d'un caractère

Nom	: PRINT	
Code d'entrée	: 04	
Paramètres d'entrée	: FCBNUM ($6244)	Numéro logique de fichier
	Accu A du 6809E	Caractère à écrire.

L'accès direct

Nom	: PUTGET	
Code d'entrée	: 07	
Les deux opérations principales d'un enregistrement sont PUT et GET.		
Paramètres d'entrée	: PUTFLG ($6249)	$00 pour GET
		$FF pour PUT
	FCBNUM ($6244) Numéro logique de fichier	
	Registre X du 6809E Numéro d'enregistrement désiré	
Paramètres de retour	: Pour PUT, le buffer est enregistré	
Paramètres de retour	: Pour GET, le buffer a été lu	

Lors de l'appel de la routine OPEN, il faut préciser la longueur des enregistrements dans RLEN ($6247-$6248), ainsi que l'adresse d'un buffer dans BUFFRE ($62AA-$62AB). Ce buffer est, bien sûr, de longueur RLEN. Le 1er enregistrement est le numéro1 Si X est nul, l'enregistrement suivant est pris (séquentiel par défaut).

Remarque	: Vous pouvez remplir ou vider le buffer vous-même, ou passer par PRINT et INPUT si vous préférez. Ceux-ci vous préviennent en cas de dépassement du buffer.

Fermeture d'un fichier

Nom	: CLOSE
Code d'entrée	: 06
Paramètre d'entrée	: FCBNUM ($6244) Numéro logique de fichier.

Lecture du catalogue

Nom	: DIR0 ou DIR1	
Codes d'entrées	: 08, 09	
Paramètres d'entrée	: DK.DRV ($6049)	Numéro de drive
	FILNAM ($624F-$6259)	Filtre 11 caractères.
Paramètres de retour	: NAMSEC ($618E) Nul, indique fin du catalogue	
	NAMSLT ($618F-$6190)	Pointe vers 32 octets:
	octets 0-7	nom
	octets 8-10	extension
	octet 11	type de fichier
	octet 12	flag ASCII binaire
	octet 13	pointeur dans la FAT
	octets 14-15	taille du dernier secteur
	octets 16-23	commentaire sur 8 octets
	octets 24-31	réservés
	FACMO ($6150-$6151) du fichier	Taille en Koctets

Le registre FILNAM sert de filtre pour le catalogue, des zéros binaires servant de joker. Si tout est à zéro, le catalogue complet est rendu. Le 1er appel se fait par DIR0, par la suite appel à DIR1. A chaque appel, DIR rend un nom de fichier.

Lecture du nom d'une disquette

Nom	: RDVOL
Code d'entrée	: 21
Paramètres d'entrée	: DK.DRV ($6049) Numéro de drive.
Paramètres de retour	: SECBUF ($6197-$6198) Pointe sur le nom (8 octets)

Backup d'une disquette

Trois routines de backup sont disponibles:
- BACKUP0 et BACKUP1 pour les TO8,TO9 et TO9+
- NEW-BACKUP pour les TO8 et TO9+ uniquement.

La différence essentielle du NEW BACKUP par rapport aux deux autres est une meilleure utilisation de l'espace mémoire. Ensuite,les pointeurs DEBZON et FINZON sont sur trois octets, le dernier étant un numéro de banque (entre 1 et NBANK $618C). DEBZON et FINZON délimitent la zone utilisée dans tous les cas.

Lecteur source différent du lecteur destination

En ce cas, DEBZON et FINZON doivent délimiter une zone mémoire d'au moins la longueur d'une piste, c'est-à-dire 2 ou 4 Ko, selon la densité du drive, sinon une erreur est générée.

Lecteur source identique au lecteur destination

Ce BACKUP utilise toute la mémoire, de la banque 1 à la banque NBANK ($618C). Vous pouvez toujours modifier NBANK pour préserver votre mémoire, le BACKUP n'en sera que plus long. Le contrôle vous est rendu lorsqu'un changement de disquette est nécessaire: à vous d'afficher le message nécessaire et de continuer le BACKUP par un appel à BACKUP1. Le flag SWPFLG vous indique une demande de SWAP. Si SWPFLG est nul, c'est que l'opération est terminée.

Fiche des routines

Pour TO8,TO9 et TO9+:

Noms	: BACKUP0 , BACKUP1
Numéros points d'entrée	: 10, 11
Pour TO8 et TO9+ uniquement :	
Nom	: NEW-BACKUP
Code d'entrée	: 71
Paramètres d'entrée	: DK.DRV ($6049) Drive destination. Accu A du 6809E Drive source. DEBZON ($616B-$616C) Zone mémoire de travail. FINZON ($616E-$616F) Fin de cette zone.
Paramètres de retour	: SWPFLG ($619D) Flag : doit-on swapper ?

Copie d'un fichier

La taille minimum du buffer est de 20 octets. Mais une taille assez grande est presque indispensable dans le cas de copie avec SWAP.

Lors de la copie,le commentaire du fichier origine est recopié si le nouveau commentaire commence par un 0. La date du fichier origine sera recopiée dans le catalogue du nouveau fichier (TO8 et TO9+ uniquement).

Comme pour BACKUP, 2 cas se présentent. Le cas d'une copie nécessitant un swap et l'inverse. Si les registres X et Y du 6809E sont égaux, EXTRAMON suppose que l'utilisateur veut copier son fichier dans le même lecteur, mais sur une autre disquette et avec le même nom. Dans ce cas, si la zone tampon n'est pas assez longue pour contenir le fichier, COPY rend la main avec SWPFLG différent de 0. C'est alors au programme appelant d'afficher un message du style "PLEASE INSERT DESTINATION etc." et lorsque l'utilisateur a changé sa disquette, il faut rappeler COPY1 qui continue le travail. C'est fini pour le programme appelant lorsque SWPFLG est nul.

Si X est différent de Y, il n'y a aucun problème et rien d'autre à faire. Attention, il faut avoir réservé au moins 2 FCBs (voir FCBINI) pour pouvoir faire COPY.

Le point d'entrée NEW-COPY (disponible uniquement sur TO8 et TO9+) est très proche de COPY0, la différence principale étant une meilleure gestion mémoire. Les pointeurs DEBZON et FINZON sont alors sur trois octets,le dernier étant un numéro de banque (entre 1 et NBANK $618C). Dans tous les cas,ces deux pointeurs délimitent la zone utilisée.

Enfin pour une copie mono lecteur, la suite se fait par appel de COPY1.

Noms	: COPY0 , COPY1
Codes d'entrée	: 12, 13
Paramètres d'entrée	: Registre X du 6809E Pointe fichier source. Registre Y du 6809E Pointe fichier destination.
Les registres X et Y pointent sur une zone de 20 octets:	
	octets 0-7 nom octets 8-10 extension octets 11-18 commentaire octet 19 numéro de drive.
	DEBZON ($616B-$616C) Zone mémoire de travail.
	FINZON ($616E-$616F) Fin de cette zone.
Paramètres de retour	: SWPFLG ($619D) Flag: doit-on swapper?

Destruction d'un fichier

Nom	: KILL
Code d'entrée	: 14
Paramètres d'entrée	: DK.DRV ($6049) Numéro de drive. FILNAM ($624F-$6259) Nom du fichier. 11 caractères. 8 octets pour le nom 3 octets pour l'extension
Paramètres de retour	: Sauf erreur, votre fichier est perdu.Entre nous, vous l'avez bien cherché.

Changement de nom d'un fichier

Nom	: NAMR
Code d'entrée	: 15
Paramètres d'entrée	: Registre X du 6809E Pointe fichier ancien nom. Registre Y du 6809E Pointe fichier prochain nom.
X comme Y pointent sur une zone de 20 octets:	
	octets 0-7 nom octets 8-10 extension octets 11-18 commentaire octet 19 numéro de drive

EXTRAMON vérifie que le nouveau nom n'est pas déjà présent sur la disquette. Lors du changement de nom, le commentaire du fichier est conservé si le nouveau commentaire commence par 0. La date du fichier est conservée (TO8 et TO9+ uniquement).

Initialisation d'une disquette

Nom	: DSKINI
Code d'entrée	: 17
Paramètres d'entrée	: DK.DRV ($6049) Numéro de disque FILNAM ($624F-$6256) Nom de volume (8octets) DK.NUM ($604D) L'entrelacement désiré (7 est bien) VERFLG ($618D) $00 pas de vérification $80 vérification (beaucoup plus lent)

Place libre sur une disquette

Nom	: DSKF
Code d'entrée	: 16
Paramètres d'entrée	: DK.DRV ($6049). Numéro de drive
Paramètres de retour	: FACMO ($6150-$6151) La place libre en Koctets

Taille d'un fichier

Nom	: LOF
Code d'entrée	: 18
Paramètres d'entrée	: FCBNUM ($6244) Numéro logique de fichier
Paramètres de retour	: FAC ($614E) Résultat en entier 16 bits (FACMO($6150)) ou en réel 4 selon la taille, pour le savoir, il faut tester VALTYP (2 ou 4).

Dans le cas des fichiers à accès direct, c'est le nombre d'enregistrements du fichier qui est rendu. En séquentiel, c'est le nombre de secteurs du fichier.

Formule de calcul pour l'accès direct :

LOF := (NBoct * NBsecteur - Inutiles) /RLEN

Avec: NBoct = Nombre d'octets par secteur
NBsecteur = Nombre de secteurs du fichier
Inutiles = Nombre d'octets inutiles du dernier secteur
RLEN = Taille d'un enregistrement

Attention: Sur le TO9,en double densité EXTRAMON prend NBoct à 256 octets au lieu de 255, d'où des petites erreurs de calcul... Ce problème n'existe plus sur TO8 et TO9+.

Numéro d'enregistrement courant

Nom	: LOC
Code d'entrée	: 19
Paramètres d'entrée	: FCB ($6244) Numéro logique de fichier
Paramètres de retour	: FAC ($6150-$6151) Résultat en entier 16 bits.
Effet	: LOC rend le numéro de l'enregistrement courant sur un fichier à accès direct. Sur un fichier à accès séquentiel, c'est le numéro du secteur courant qui est rendu.

Exemple d'utilisation

Le programme ci-dessous représente une application simple du DOS en assembleur. Son but est d'afficher à l'écran le nom d'une disquette :

```
        ORG     $A000
EXTRA   EQU     $EC0C
PUTC    EQU     $E803
SECBUF  EQU     $6197
DK.DRV  EQU     $6049
RDVOL   EQU     21
        LDX     #NBUF       Initialisation du pointeur.
        STX     SECBUF      SECBUF sur NBUF.
        LDB     #01         Initialisation d'EXTRAMON.
        JSR     EXTRA       -
        CLRA
        STA     DK.DRV      Numéro de drive 0
        LDB     #RDVOL      Recherche du nom de disque.
        JSR     EXTRA       -
        LDX     #NOM        Affichage.
REC     LDB     ,X+         -
        JSR     PUTC        -
        CMPX    #FBUF
        BNE     REC         -
        SWI
NOM     FCC     /NOM DU DISQUE:/
NBUF    RMB     8
FBUF    EQU     *
        END
```

6. L'éditeur

Cette routine permet d'utiliser un éditeur ayant toutes les fonctionnalités de l'éditeur BASIC (insertion, effacement, déplacement curseur, etc.). Il fonctionne aussi bien en mode TO7-70 qu'en mode 80 colonnes. L'édition se fait dans la fenêtre courante, nous vous conseillons de vous reporter à l'étude de la routine PUTC du moniteur et sur ses séquences d'échappements US pour fixer cette fenêtre (page 175)

En entrée l'appelant fournit un buffer, en sortie il reçoit la ligne lue. L'éditeur teste (entre autres) l'octet IWTFLG. Dans le cas où ce dernier est différent de zéro, on sort de l'éditeur avec un tampon vide. Ceci permet quelques fantaisies comme par exemple reprendre la main sur une touche fonction ! Pour ce faire,le registre IWTFLG peut être modifié dans une routine d'interruption ou dans l'indirection de GETC du moniteur.

Il est également possible de demander à l'éditeur de prendre un texte pointé par DEFTXT à la place de l'entrée clavier, le texte devant se terminer par $00. Pour ne pas avoir de texte par défaut, il est recommandé de pointer DEFTXT sur un $00.

A titre d'exemple, citons l'instruction AUTO du BASIC qui fonctionne selon ce principe. En entrant dans l'éditeur, DEFTXT pointe sur le numéro de ligne à afficher; ainsi ce numéro de ligne est écrit à l'écran comme si l'utilisateur l'avait frappé.

Comme nous le disions quelques lignes auparavant, grâce à l'indirection de GETC, l'application peut dériver DEFTXT sur un texte lors d'un appui sur une touche fonction (F1 déclenche un RUN, F2 un LIST, etc.). En fait, il faut que la routine indirectée de GETC rende le 1er caractère "L" et fasse pointer DEFTXT vers LST,$00. Si par exemple, il y a un CR dans la ligne, elle est automatiquement validée par l'éditeur.

Nom	: EDIT
Code d'entrée	: 22
Paramètres d'entrée	: Registre X du 6809E Adresse du tampon Registre Y du 6809E Fin du tampon (dernier octet utile) DEFTXT ($61E0-$61E1) Pointe sur un texte se substituant à l'entrée clavier. IWTFLG ($62A9) Sémaphore de sortie rapide
Paramètre de sortie	: Néant
Effet	: Le tampon est rempli lors de la frappe de la touche ENTREE. La fin de la ligne se reconnaît par un $00. Les accents sont codés avec les séquences SS2. Les caractères non reconnus par l'éditeur sont effacés à l'écran. Si l'utilisateur frappe CNT-C, EDIT rend la main à l'application avec un tampon vide.
Remarque	: Il ne faut pas fournir de tampon plus petit que 3 caractères.

7. L'interpréteur musical

La routine PLAY est à l'extramoniteur ce que l'instruction PLAY est au BASIC. C'est dire sa facilité d'utilisation ! Après avoir précisé dans l'accumulateur A la longueur de la chaine à jouer et pointé par le registre Y l'adresse où elle se trouve, il suffit d'appeller la routine PLAY pour que vos oreilles mélomanes jouissent de vos talents de compositeur. La syntaxe de la chaîne à interpréter, son contenu, ses paramètres ainsi que ses valeurs par défaut sont rigoureusement les mêmes que pour l'instruction PLAY du BASIC (Octave, durée, tempo, etc.)

```
Nom                    : PLAY
Code d'entrée          : 23
Paramètres d'entrée    : Registre Y du 6809.E
                         Accumulateur A du 6809.B
Paramètres de retour   : Néant
Effet                  : Exécute la mélodie désignée par Y

Exemple

* Balayage de la gamme de DO octave 3 a
* DO octave 5 (dernière note est ronde)

       TITLE  EXMUSIC
       ORG    $A000
EXTRA  EQU    $EC0C
RESETW EQU    01
PLAY   EQU    23

FLAG   EQU    *        RESET GENERAL
       LDB    #RESETW
       JSR    EXTRA

FLAG1  EQU    *        prog etudie
       LDA    #39      longeur chaine
       LDY    #MUSIC
       LDB    #PLAY
       JSR    EXTRA
       SWI

MUSIC  FCC    /O3DOREMIFASOLASI/
       FCC    /O4DOREMIFASOLASI/
       FCC    /L9GO5DO/

       END
```


8. Les messages d'erreur en anglais

Les messages d'erreur générés sous BASIC représentent une suite de chaînes de caractères qui peuvent être appelés par l'extramoniteur. La procédure est la suivante:

Tout d'abord on implante dans l'accumulateur A le numéro du message d'erreur dont on désire l'affichage (par exemple 02 pour "Syntax Error").

– Ensuite il faut pointer par le registre d'index X un buffer où sera rangée la chaîne de caractères correspondante.

– Enfin vous appelez la routine ERRMSG qui implantera le message dans le buffer pointé.

Notez que la routine ne se charge pas de l'affichage. Ainsi, dans l'exemple proposé ci-dessous nous utilisons la routine PUTC ("Affichage des caractères alphanumériques, page 163) dans une boucle, afin de transférer sur l'écran le contenu du buffer.

```
Nom                    : ERRMSG
Code d'entrée          : 20
Paramètres d'entrée    : Accumulateur A du 6809.E
                         Registre X du 6809.E
Paramètres de retour   : Registre X du 6809.E
Effet                  : Transfert dans un buffer pointé par X le message d'erreur
                         repéré par A.
Exemple                : Le programme ci-dessous vous offrira un spectacle que
                         j'espère original pour vous, puisqu'il affiche une
                         trentaine de messages d'erreurs simultanément !

* Affichage des 28 derniers messages
* d'erreurs du basic (du no 50 au 78)

       TITLE  EXMESS
       ORG    $A000
EXTRA  EQU    $EC0C
PUTC   EQU    $E803
RESETW EQU    01
ERRMSG EQU    20

FLAG   EQU    *        RESET GENERAL
       LDB    #RESETW
       JSR    EXTRA
```

```
FLAG1   EQU     *       prog etudie
        LDA     #50     message numero 50
AUTRES  LDX     #$8000  implant tampon
        LDB     #ERRMSG routine numero 20
        JSR     EXTRA
AFICHE  LDB     ,X+     affichage du.
        JSR     PUTC    tampon par PUTC
        CMPB    #$00    test fin de chaine
        BNE     AFICHE
        INCA
        CMPA    #79     test dernier mess
        BNE     AUTRES  message suivant
        SWI
        END
```

Note : Les numéros des messages d'erreurs BASIC sont répertoriés dans les guides accompagnant les unités centrales.

Le DOS iconique

Généralités

Le DOS Iconique, appelé sur les menus "Exploitation de fichiers", est situé sur la banque 3 du slot 0 des TO8, TO9 et TO9+. Il utilise pour réaliser ses différentes manipulations de fichiers, des routines du moniteur et de l'extra-moniteur. Ainsi, globalement, ses variables se situent dans les pages $60, $61 et $62, ce qui implique que toute application utilisant le DOS Iconique doit laisser ces 3 pages libres.

Comme toutes routines de l'extramoniteur, le début du programme sera consacré aux initialisations des sous-ensembles concernés:
- Unités de disquettes
- Fenêtre graphique (nulle par défaut)
- Curseur graphique qui définit la position des coins gauches des tableaux générés par le DOS Iconique.
- Mise à jour du registre COULEUR ($619F)
- Le registre CHDRAW ($6041) doit être à 0.

Le DOS Iconique ne sauvegarde pas l'écran avant d'afficher ses fenêtres, il faudra donc prévoir un traitement adéquat dans vos programmes personnels.

Trois routines sont utilisables par une application externe:
- La sélection de fichiers,
- La saisie d'un nom de fichier,
- La sélection du lecteur courant.

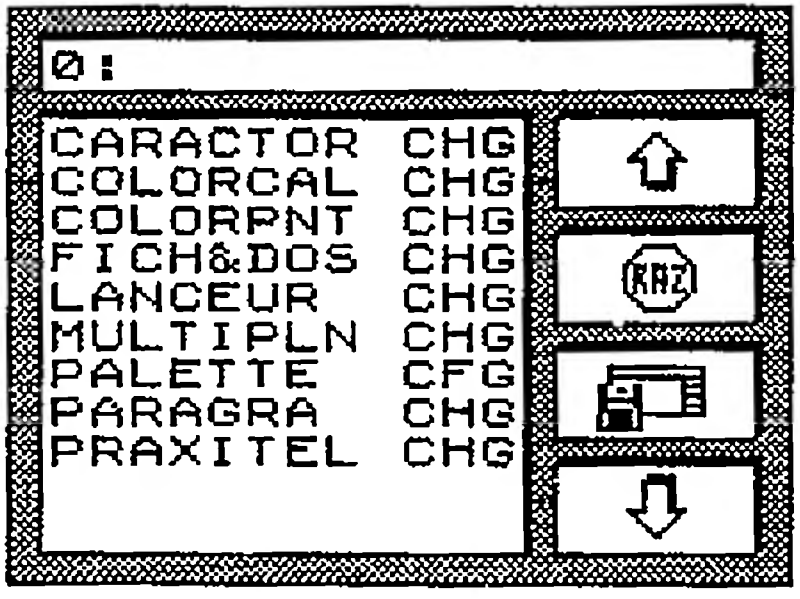
Ainsi, l'utilisateur peut à l'aide du crayon optique ou de la souris (si elle est en action), sélectionner un fichier, relire le catalogue et se déplacer à l'intérieur. Toutes les commandes light pen ou souris sont doublées au clavier par les touches RAZ, ENTREE, les chiffres de 0 à 4 et les flèches verticales.

Pour réaliser ces tâches, un buffer pointé par le registre d'index X doit être alloué, sa longueur étant définie par le registre Y. Si ce buffer est trop petit, une erreur "Out of Memory" sera générée.

Sélection de fichiers

La routine DIRR fournit un catalogue réduit (nom + extension), classé alphabétiquement, de la disquette placée dans le lecteur courant. Au préalable, les registres DK.DRV et FILNAM indiqueront respectivement le numéro de lecteur concerné et le filtre de sélection des noms de fichiers (0 étant le caractère transparent). En sortie, le registre B retourne le numéro de l'erreur. Si son contenu vaut 0, aucune erreur n'est détectée et le nom est inscrit dans le registre FILNAM.

Un buffer de 1500 octets est nécessaire pour accomplir cette routine, sachant que la fenêtre affichée à l'écran a une taille de 20 × 15 caractères.

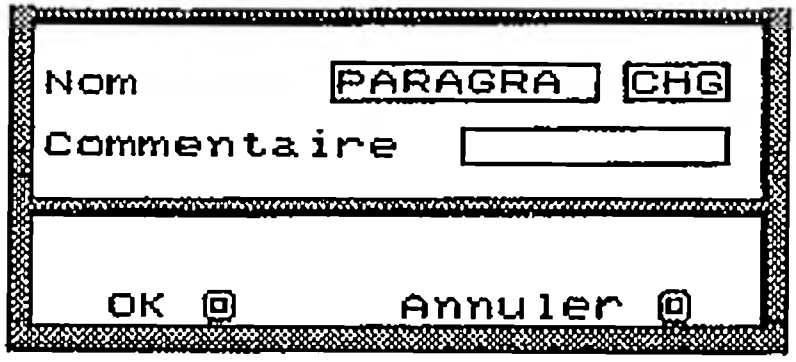


Saisie d'un nom de fichier

La routine SAISIE permet la frappe d'un nom de fichier complet constitué du nom, de l'extension et du commentaire. Le registre FILNAM contient le nom par défaut proposé à l'utilisateur suivi de l'extension et du commentaire (8+3+8 caractères).

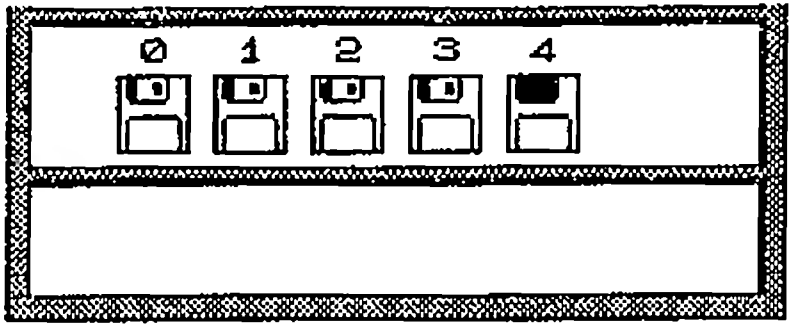
La saisie est validée par le pointage de OK dans le tableau affiché ou par la frappe de la touche ENTREE. La touche RAZ ou le pointage de "Annuler" interrompt la saisie.

Un buffer de 100 octets est nécessaire pour cette routine. Le tableau affiché occupe 24 × 11 caractères.



Sélection du lecteur courant

La routine SELDEV permet de sélectionner le lecteur courant. Le registre DK.DRV contient le numéro du lecteur à mettre en évidence. Un buffer de 100 octets est nécessaire. La taille de la fenêtre affichée est de 24 × 11 caractères.



Appel au DOS Iconique

L'appel de l'une des routines décrites ci-dessus se fait via la routine COMS du moniteur. Pour de plus amples détails, veuillez vous reporter au chapitre "Commutation des mémoires ROM", page 222.

Exemple:

```
        LDX     #$B000      Début du buffer
        LDY     #1700       Longueur de la zone (1700 octets)
        LDA     #03         Appel du slot 0, banque 3
        LDU     #ROUTINE    Point d'entrée de la routine ex $3FC1
                            pour DIRR
        JSR     COMS
        TSTB
        BNE     ERROR
OK      EQU                 *
```

Il conviendra d'ajouter au début du programme les initialisations des registres concernés par la routine appelée. Vous trouverez ci-dessous un résumé des equates du DOS Iconique.

```
DK.DRV  EQU     $6049       Numéro de lecteur courant
SECBUF  EQU     $6197       Pointer de buffer disque
FATPTR  EQU     $6199       Pointer de buffer FAT
FILNAM  EQU     $624F       Nom de fichier courant
CHDRAW  EQU     $6041       Caractère graphique
COULEUR EQU     $619F       Couleur courante
EXTRA   EQU     $EC0C       Point d'entrée d'extramon
COMS    EQU     $EC03       Point d'entrée de commutation de slot

DIRR    EQU     $3FC1       Point d'entrée du catalogue réduit
SAISIE  EQU     $3FC4       Point d'entrée de SAISIE
SELDEV  EQU     $3FC7       Point d'entrée de SELDEV
```


10. Informations complémentaires

Extramon sous BASIC 512

Le BASIC 512 disponible sur les TO8 et TO9+, initialise automatiquement l'extramoniteur. En conséquence, vos programmes personnels appelant extramon à partir du BASIC 512 n'ont pas à realiser ce traitement. Si vous désirez récupérer les erreurs éventuelles, il faut rediriger le vecteur de rattrapage d'erreur ZPERR ($6185) de la manière suivante:

```
SAVSTK  EQU     $6175
        LDS     SAVSTK
        PULS    A,DP,X,Y,U,PC
        .
        .
```

Dans le cas contraire, c'est le message d'erreur BASIC qui sera affiché.

Les numéros de fonctions ou routines d'extramon

```
RESETC  EQU     0       reset à froid
RESETW  EQU     1       reset à chaud
FCBINI  EQU     2       init d'un FCB
OPEN    EQU     3       ouverture
PRINT   EQU     4       output
INPUT   EQU     5       input
CLOSE   EQU     6       close
PUTGET  EQU     7       accès direct
DIR0    EQU     8       catalogue
DIR1    EQU     9       catalogue suite
BACKUP0 EQU     10      backup
BACKUP1 EQU     11      backup suite
COPY0   EQU     12      copie
COPY1   EQU     13      copie suite
KILL    EQU     14      destruction d'un fichier
NAME    EQU     15      renomme un fichier
DSKF    EQU     16      calcul place libre
DSKINI  EQU     17      initialise la disquette
LOF     EQU     18      taille d'un fichier
LOC     EQU     19      adresse écriture dans fichier
```

ERRMSG	EQU	20	rend erreur en clair
RDVOL	EQU	21	va lire le nom du disque
EDIT	EQU	22	l'éditeur plein écran
PLAY	EQU	23	l'interpréteur musical
CIRCLE	EQU	24	dessin du cercle ou ellipse
PSETXY	EQU	25	écrit un point
LINE	EQU	26	trace une ligne
BOX	EQU	27	trace un rectangle
CHOIX	EQU	28	initialisation mode graphique
PAINT	EQU	29	remplissage d'une surface
MIG	EQU	30	l'interpréteur graphique
TRACE	EQU	31	tracé de tortue
ANIME	EQU	32	liberté de la tortue
SHOW	EQU	33	allumage de la tortue
HEAD	EQU	34	direction de la tortue
ROT	EQU	35	rotation de la tortue
ZOOM	EQU	36	taille de la tortue
FWD	EQU	37	avance de la tortue
MOVE	EQU	38	déplacement de la tortue
INITORTUE	EQU	39	initialise une tortue
CMPTORTUE	EQU	40	compile une forme de tortue
SGN	EQU	41	signe
INT	EQU	42	entier
ABS	EQU	43	valeur absolue
SQR	EQU	44	racine carrée
LOG	EQU	45	logarithme népérien
EXP	EQU	46	exponentielle
COS	EQU	47	cosinus
SIN	EQU	48	sinus
TAN	EQU	49	tangente
FRCTYP	EQU	50	force le type
FIXER	EQU	51	troncature
FRND	EQU	52	valeur aléatoire
NEGGO	EQU	53	négation
ADDGO	EQU	54	addition
SUBGO	EQU	55	soustraction
MULTGO	EQU	56	multiplication
DIVGO	EQU	57	division réelle
EXPGO	EQU	58	exponentiation
IMODO	EQU	59	reste de la division entière
IDIVO	EQU	60	division entière
MOVFM	EQU	62	FAC := mémoire
MOVMF	EQU	63	mémoire := FAC
MOVAF	EQU	64	ARG := FAC
FIN	EQU	65	conversion ASCII = = > binaire
PUFOUT	EQU	66	C.binaire = = > ASCII décimal
HOFOUT	EQU	67	C.binaire = = > ASCII héxa/octal

*POUR TO8 ET TO9+ UNIQUEMENT:

ATN	EQU	68	arctangente
CODE	EQU	69	code et décode image
INICACHE	EQU	70	cache disque
NEWBACKUP	EQU	71	backup
NEWCOPY	EQU	72	copie

Les equates d'extramon

***** MODES D'OUVERTURE POUR LA ROUTINE OPEN *****

M.SQI	EQU	$10	ouverture en input séquentiel
M.SQO	EQU	$20	ouverture en output séquentiel
M.RND	EQU	$40	ouverture en direct (I/O)

***** LES OBJETS TORTUES **********

TX	EQU	6	position de la tortue en X
TY	EQU	9	position de la tortue en Y
TROT	EQU	12	rotation de la tortue
TTAI	EQU	13	taille de la tortue
TDIR	EQU	14	direction de la tortue
TFORME	EQU	16	forme de la tortue

***** L'ACCUMULATEUR FLOTTANT **********

DBLFLG	EQU	$6103	flag de double précision
VALTYP	EQU	$6105	indicateur de type
FAC	EQU	$614E	accumulateur
FACEXP	EQU	$614E	exposant
FACHO	EQU	$614F	octet fort de la mantisse
FACMO	EQU	$6150	octet moyen de la mantisse
FACLO	EQU	$6151	octet faible de la mantisse
DFACHO	EQU	$6152	4 octets de plus pour double précision
DFACMH	EQU	$6153	
DFACML	EQU	$6154	
DFACLO	EQU	$6155	
FACSGN	EQU	$6156	signe de FAC (0 ou -1) quand non codé

```
*****LES ARGUMENTS FLOTTANTS (NON CODES)*****************

ARGEXP      EQU     $6159
ARGHO       EQU     $615A
ARGMO       EQU     $615B
ARGLO       EQU     $615C

DARGHO      EQU     $615D       4 octets de plus pour double précision
DARGMH      EQU     $615E
DARGML      EQU     $615F
DARGLO      EQU     $6160

ARGSGN      EQU     $6161

*
DEBZON      EQU     $616B       Début et fin de zone tampon pour
FINZON      EQU     $616E       EXTRAMON

EOFFLG      EQU     $6178       flag fin de fichier
                                zéro - non fini
                                non zéro - fini

***** TEMPORAIRES POUR PRINT USING**********************

DFWID       EQU     $617A
FLDWID      EQU     $617B
PUMASK      EQU     $617C

ZPERR       EQU     $6185       rattrapage d'erreur d'EXTRAMON
                                BASIC l'initialise à ERROR dans
                                CLEAR

RSKCHG      EQU     $6189       Flag de risque de changement de
                                disque

NBANK       EQU     $618C       nombre de banques accessibles

***** VARIABLES LIEES AUX DISQUES **********************

VERFLG      EQU     $618D       pour VERIFY ON ou OFF
NAMSEC      EQU     $618E       le secteur du catalogue contenant le
                                nom recherché.

NAMSLT      EQU     $618F       un pointeur vers le slot dans le
                                secteur (voir NAMSEC), ou zéro si
                                le nom n'a pas été trouvé.
```



```
CARCOU      EQU     $6196       Le caractère que DFCHRI vient de
                                lire.

                                SECBUF et FATPTR doivent être
                                positionnés avant le 1er appel au
                                DOS sous peine d'erreur.

SECBUF      EQU     $6197       Un pointeur vers le buffer de lecture
                                d'un secteur

FATPTR      EQU     $6199       pointeur vers la zone où l'utilisateur
                                veut ranger les FATs (DSBs)

DSBLEN      EQU     6+2*80

SWPFLG      EQU     $619D       un flag pour dire que l'on veut un
                                swap disquette.

***** VARIABLES GLOBALES NECESSAIRES AU GRAPHIQUE ******

COULEUR     EQU     $619F
TRATYP      EQU     $61A0       type de tracé  0 pour normal
                                               1 normal sans couleur
                                               2 en ou exclusif
XXXX        EQU     $61A1
YYYY        EQU     $61A3       le curseur graphique.

XL          EQU     $61A5       La fenêtre graphique
YB          EQU     $61A7
XR          EQU     $61A9       de XLeft, YBottom
YT          EQU     $61AB       à XRight, YTop.

***** TEMPORAIRE POUR L'EDITEUR **************************

DEFTXT      EQU     $61E0       pointeur vers du texte à afficher

***** VARIABLES LOCALES AU TRACE D'ELLIPSES **************

FILFLG      EQU     $61EF       doit-on remplir l'ellipse ou le
                                rectangle...
AXEV        EQU     $61F0       axe vertical.
AXEH        EQU     $61F1       axe horizontal.
CAMFLG      EQU     $61F2       camembert ?
ALPHA1      EQU     $61F3
ALPHA2      EQU     $61F7
```

```
***** VARIABLES LOCALES AU CODAGE ET AU DECODAGE
                                  *********

XOCOD     EQU   $61D6   extrémités du rectangle
YOCOD     EQU   $61D7
X1COD     EQU   $61D8
Y1COD     EQU   $61D9


***** DIVERS de $6200 à $62FF ****************************.

TYPDSK    EQU   $6219   Type de contrôleur
TABDEN    EQU   $621A   table des densités par disque
MAXMOD    EQU   $6223
FCBNUM    EQU   $6244   numéro du FCB courant
RLEN      EQU   $6247   longueur de l'enregistrement du
                        fichier à accès direct.
PUTFLG    EQU   $6249   pour distinguer PUT de GET !
FILMOD    EQU   $624B   mode du fichier (OPEN).
FILTYP    EQU   $624C   type de fichier.
                        0 = BASIC program.
                        1 = BASIC data file.
                        2 = machine language file.
ASCFLG    EQU   $624D   Flag ASCII.
                        0 = fichier non ASCII.
                        FF = fichier ASCII.
FILNAM    EQU   $624F   buffer nom de fichier.
FILEXT    EQU   $6257   buffer extension nom de fichier
OPTBUF    EQU   $625A   buffer de descriptions des options.
MACP      EQU   $627D   le pointeur vers le motif de peinture.
WITH      EQU   $6288   flag avec ou sans couleur.

IWTFLG    EQU   $62A9   pour l'éditeur, lorsque ce flag devient
                        <> de 0, on sort sans réfléchir !!!
BUFFRE    EQU   $62AA   pour l'OPEN en accès direct.
                        Pointe vers un buffer (de longueur
                        RLEN) où l'appelant retrouvera
                        l'enregistrement demandé.

ONOFF     EQU   $62B0   Flag avec ou sans disque
BLOCS     EQU   $62AE   Pointe sur une table de descripteur de
                        blocs
```

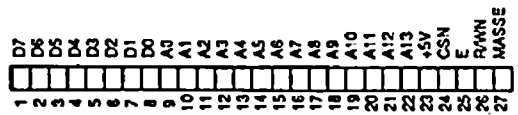

Annexe

Connectique

Les figures suivantes décrivent les interconnexions accessibles à l'utilisateur par l'intermédiaire des prises et fiches des unités centrales TO9, TO8, TO9+.

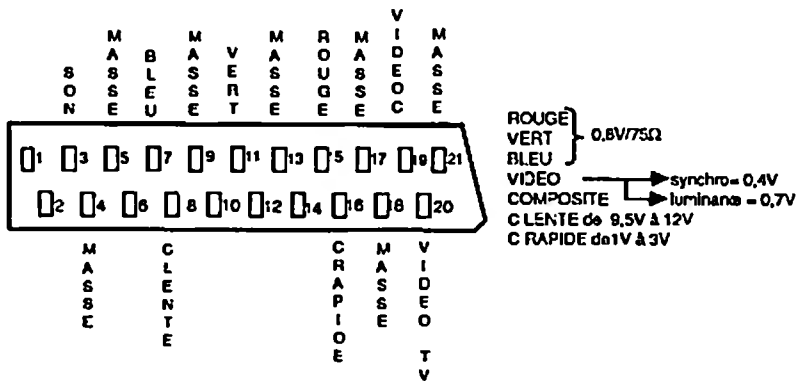
Chaque dessin correspond à une vue de face du connecteur, côté utilisation.

Connecteur cartouche TO9, TO8, TO9+

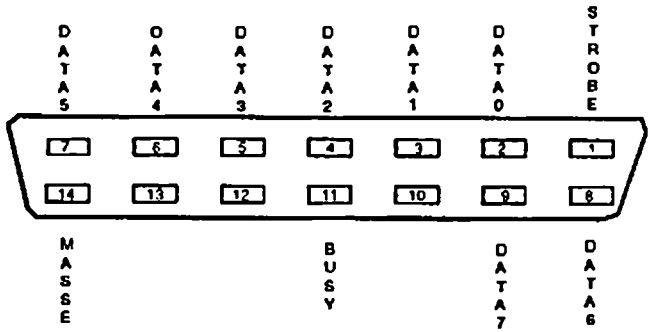


Tous les signaux de ce connecteur sont compatibles TTL

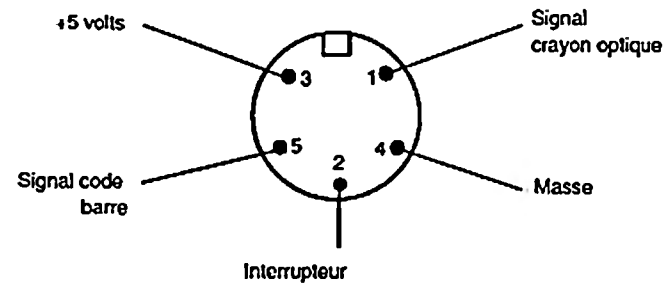
Prise péritel TO9, TO8, TO9+



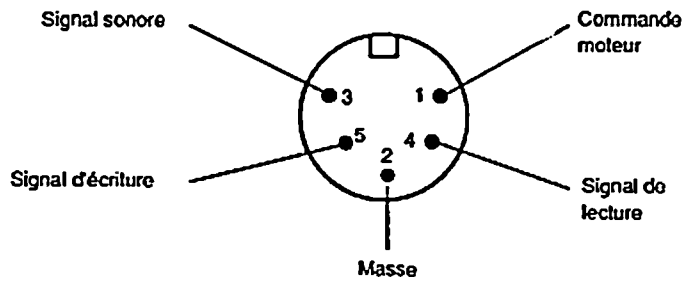
Connecteur imprimante type "Centronics" TO9, TO8, TO9+



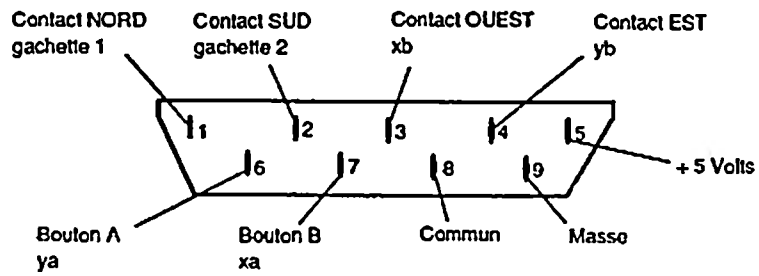
Connecteur crayon optique TO9, TO8, TO9+



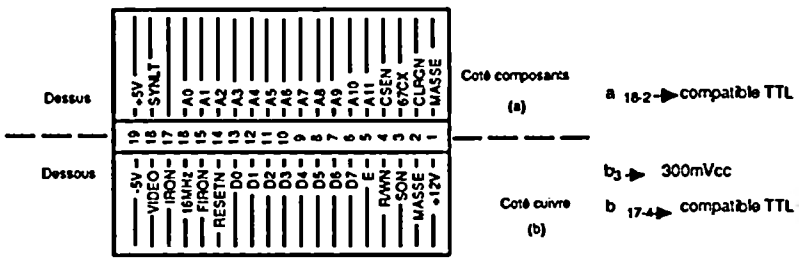
Connecteur magnétophone TO9, TO8, TO9+



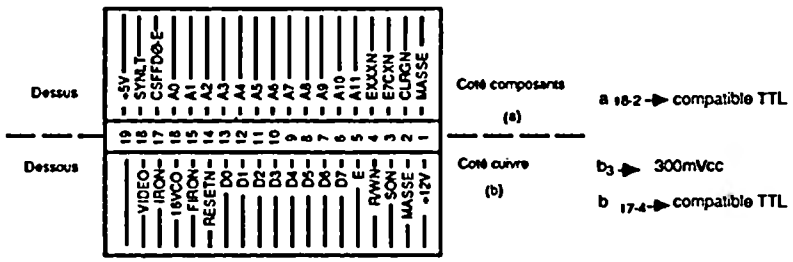
Connecteur manettes de jeux/souris TO9, TO8, TO9+



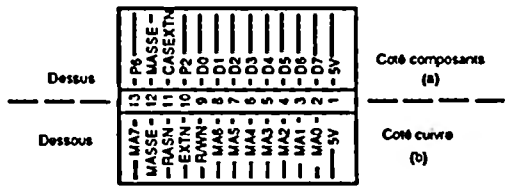
Connecteur extension (contrôleurs) TO9



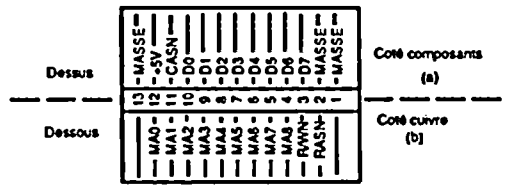
Connecteur polyvalent ou universel TO8, TO9+



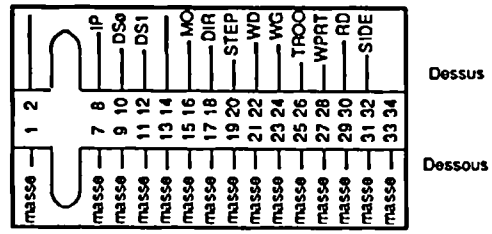
Connecteur extension disque virtuel TO9



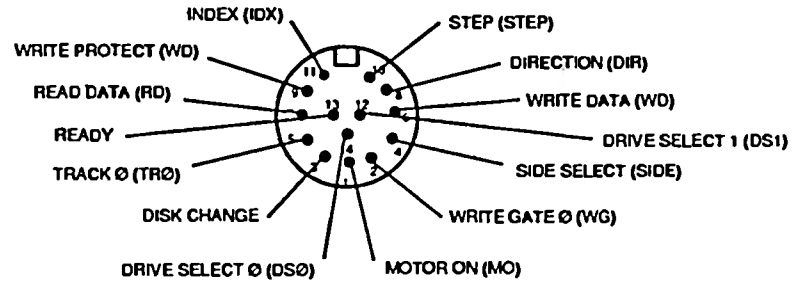
Connecteur extension mémoire TO8



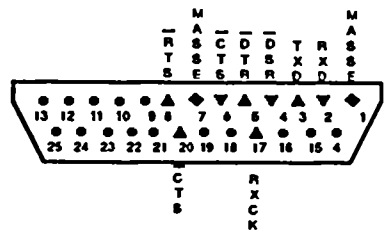
Connecteur drive externe TO9



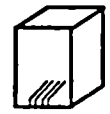
CONNECTEUR DE LECTEUR ENREGISTREUR
DE DISQUETTES ET QDD TO8, TO9+



Connecteur RS 232 TO9+



Connecteur de raccordement à la ligne PTT (TO9+)



T2, T1, L2, L1